PROTECTING CYBER-PHYSICAL SYSTEMS WITH SPECIAL EMPHASIS ON BUILDING AUTOMATION NETWORKS

A Dissertation

by

ZHIYUAN ZHENG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	A. L. Narasimha Reddy
Co-Chair of Committee,	Hussein Alnuweiri
Committee Members,	Riccardo Bettati
	P. R. Kumar
	Le Xie
Head of Department,	Miroslav M. Begovic

December 2017

Major Subject: Computer Engineering

ABSTRACT

Critical infrastructures such as power grids, water treatment and distribution facilities, and Building Automation Systems (BAS) have come to employ Cyber-Physical Systems (CPSs) in which physical devices or components are coordinated and controlled through communication networks. Due to the criticalness of the infrastructures in which CPSs are deployed, they have become a ripe target for cyber-attacks. This work focuses on developing solutions to protect CPSs from cyber-attacks.

To understand the network traffic behavior in a CPS, a collection of BACnet traffic was collected from a real-world BAS network. We conducted in-depth traffic analysis and observed that BACnet traffic can be classified into three categories: Time-driven, Human-driven, and Event-driven. Based on the observed traffic behavior, we developed "THE-driven" anomaly detector which adopts different mechanisms for each category of traffic. In addition, Commensurate Response (CR) was introduced to improve the system resilience and attack survivability of the CPS. CR forces the footprint of the attack to be commensurate with its impact on the system. Next, Path Redundancy was proposed to counter compromised embedded controllers which could be leveraged by attackers to launch data integrity attacks and false command attacks. As an extension of Path Redundancy, a new CPS architecture is introduced to provide data replica and enable control switching when a controller is attacked. The new architecture leverages virtualization to overcome Single-Point-of-Failures (SPOFs) without requiring additional hardware devices.

DEDICATION

To my dear parents and many other important people in my life.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Professor A. L. Narasimha Reddy, for his tremendous guidance, support and encouragement during the course of my doctoral study. His insights into research, pursuit of excellence, and attitude of life have inspired me significantly. I feel very grateful to be his student and work with him over the years.

I would like to thank Professor Riccardo Bettati for his guidance and help with my research. The detailed and in-depth discussion with him have invaluably shaped the course of this work.

I would like to thank Professors Hussein Alnuweiri, P. R. Kumar, and Le Xie for their insightful comments and feedback on this work.

I would also like to thank Carl Neilson and David Fisher who provided insight and expertise to help me understand the problem.

I am also grateful to all my friends and colleagues for their support during my graduate student life. Special thanks to Allen Webb for being an amazing collaborator and friend and providing tremendous help in my research. I would also like to thank Ya Wang, Jingwei Xu, Ke Ma, Ye Wang, Yingyezhe Jin, and Hanbin Hu for their wonderful friendship that makes my life in College Station memorable.

Finally, I would like to thank my parents and family for their unconditional love, support and encouragement that have enlightened and enriched my life.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors A. L. Narasimha Reddy, Hussein Alnuweiri, P. R. Kumar, and Le Xie of the Department of Electrical & Computer Engineering; and Professor Riccardo Bettati of the Department of Computer Science & Engineering.

The network traffic dataset for Section 2 was collected with the help of David Brown and Christopher Dieckert. Section 3 is based on a paper jointly written with Shan Jin and Professor Riccardo Bettati of the Department of Computer Science & Engineering.

All other work conducted for the dissertation was completed by the student.

Funding Sources

Graduate study and research was supported by the National Science Foundation under Grant no. CNS 1218929, the Qatar National Research Foundation 9th Cycle under Grant no. 9-069-1-018, and a seed grant from Texas A&M Engineering Experiment Station (TEES).

NOMENCLATURE

ACI	Auxiliary Controller Instance
ARCNET	Attached Resource Computer NETwork
ARP	Address Resolution Protocol
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
AUC	Area Under the Curve
BACnet	A Data Communication Protocol for Building Automation and Control Networks
BAS	Building Automation System
BBMD	BACnet/IP Broadcast Management Device
BLN	Building Level Network
C&C	Command and Control
CAA	ConfirmedAcknowledgeAlarm
CAN	Controller Area Network
CCOV	ConfirmedCOVNotification
CEN	ConfirmedEventNotification
CF	CPU Factor
CPS	Cyber-Physical System
CPT	ConfirmedPrivateTransfer
CPU	Central Processing Unit
CR	Commensurate Response
DNP3	Distributed Network Protocol version 3
DoS	Denial-of-Service

DPI	Deep Packet Inspection
ECU	Electronic Computing Unit
FDIA	False Data Injection Attack
FFT	Fast Fourier Transform
FLN	Field Level Network
FP	Field Panel
HMI	Human Machine Interface
HVAC	Heating, Ventilation, Air Conditioning
IAM	I-AM
IDS	Intrusion Detection System
IHAVE	I-HAVE
IP	Internet Protocol
LF	Link Factor
LTI	Linear Time-Invariant
MF	Memory Factor
MITM	Man-in-the-Middle
MMS	Multimedia Messaging Service
MPLS	Multi-Protocol Label Switching
MS/TP	Master-Slave/Token-Passing
MTU	Master Terminal Units
NS-2	Network Simulator 2
ONT	Object Name Table
OSI	Open Systems Interconnection
P2P	Peer-to-Peer

PCI	Primary Controller Instance
PD	Physical Device
PID	Proportional–Integral–Derivative
PLC	Programmable Logic Controller
QoS	Quality of Service
RAM	Random-Access Memory
RD	Replication Degree
RIO	Remote I/O
ROC	Receiver Operating Characteristic
RP	ReadProperty
RPM	ReadPropertyMultiple
RR	ReadRange
RTT	Round-Trip Time
RTU	Remote Terminal Units
RWOU	Response-without-Update
RWU	Response-with-Update
SCADA	Supervisory Control and Data Acquisition
SCOV	SubscribeCOV
SNMP	Simple Network Management Protocol
SPOF	Single-Point-of-Failure
TCP	Transmission Control Protocol
TEC	Terminal Equipment Controllers
TF	Timing Factor
TRC	Time Regularity Classifier

TS	TimeSynchronization
UDP	User Datagram Protocol
VD	Virtualization Degree
VPLC	Virtual Programmable Logic Controller
VPN	Virtual Private Networks
VTEC	Virtual Terminal Equipment Controllers
WAN	Wide Area Network
WHOHAS	WHO-HAS
WHOIS	WHO-IS
WP	WriteProperty
WPM	WritePropertyMultiple

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Cyber-physical Systems (CPSs) have become the core components of safety-critical infrastructures such as smart grid [1], building automation networks [2] and water/sewage plants [3]. CPSs bridge the cyber and the physical worlds by integrating the computing and communication capabilities of the former to monitor, model, control, and analyze processes in the latter. The communication networks employed in such Supervisory Control and Data Acquisition Systems are normally termed SCADA networks. A number of different protocols such as DNP3 [4], Modbus [5], BACnet [6, 7], and other protocols are employed in these networks. Often, the security protocols deployed in such systems are not corresponding with their criticality.

Due to the criticalness of the infrastructures in which CPSs are deployed, they have become a ripe target for cyber-attacks. Examples include Stuxnet [8], which attacked Iran's nuclear centrifuges by sending malicious commands to periodically modify their motor's spinning frequency while reporting normal sensing values. Similarly, hackers caused the massive Ukraine power outage [9] on Dec. 23rd, 2015, by sabotaging the control system and remotely opening the breakers. This incident affected about 230,000 people and was regarded as the first power outage caused by hackers.

This work focuses on developing solutions to protect CPSs from cyber-attacks, with a focus on Building Automation Networks. We assume attackers are able to penetrate into the CPS network and compromise embedded controllers. The solutions in this dissertation protect the CPS from different perspectives: identifying malicious CPS network traffic using anomaly detection; improving system resilience and attack survivability of CPS through Commensurate Response; improving data validity through Path Redundancy; and introducing a new CPS architecture to overcome the Single-Point-of-Failures.

First, we understand the network traffic of a CPS network and develop an anomaly detector. We collect BACnet traffic from a real-world Building Automation Network for several days, and analyze the traffic behavior at the aggregated level and the individual flow-service level. We observe that the BACnet traffic is a combination of multiple flow-service streams that belong to "THE-driven" categories: Time-driven, Human-driven, and Event-driven. Time-driven traffic follows periodic patterns, regular patterns, or on/off models. Human-driven and event-driven traffic present non-periodic patterns. Such traffic behavior has not been observed in previous CPS networks. Based on the traffic patterns, we construct flow-service models for the time-driven traffic and develop THE-Driven Anomaly Detector which adopts different mechanisms for each category of traffic. We evaluate the anomaly detector using k-fold cross validation and synthetic attacks. The proposed THE-Driven Anomaly Detector is shown to be able to effectively detect suspicious traffic in BAS networks with a small false alarm rate.

Second, current protocols and applications in CPSs allow significant changes to a system to take place within a short time or small network footprint, which can be exploited by attackers to cause a great impact on the physical systems. The challenge is how can we minimize the impact of a potential attack so that the system can continue to operate safely. We introduce adaptive Commensurate Response (CR) to narrow down the asymmetry between the cost of attacks and their impact through enforcing command footprints to be commensurate with their impact on the system. Such impact is measured by the change of the setpoint (change-driven CR) or the distance between the operating state and the critical state (criticality-driven CR). A larger change of the setpoint corresponds to a larger response time; also, the system slows down the response as it approaches the critical state. Change-driven CR can be used against setpoint attacks with big setpoint changes, and criticality-driven CR is effective to combat both setpoint attacks and actuation attacks. Detailed analysis and design are provided in Section 3. We also conduct a case study on

automobile cruise control and demonstrate that CR can effectively improve the system resilience and attack survivability while satisfying QoS requirements.

The current CPS architecture follows a hierarchical tree structure. It employs distributed and local Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs) to control and monitor different types of Physical Devices (PDs) such as sensors, valves, pumps, drives, boilers and generators. Typically, each PD is only connected with one PLC or RTU. If a PLC is compromised, the attackers could directly send malicious commands to the connected PDs, causing unexpected changes to the physical system. They could also deceive the SCADA Server with invalid sensing values, compromising the data integrity of the CPS networks. The embedded controllers (PLCs and RTUs) become the Single-Point-of-Failures (SPOFs) of the SCADA networks.

In Section 4, we explore the path diversity in the communication channel and propose Path Redundancy to allow redundant sensing paths between the Server and a PD. The redundant paths go through other existing PLCs and are only used for sensing (data acquisition) purposes. The redundant sensing values are used to validate the data collected from the original path. Path Redundancy can prevent data integrity attacks (sending false sensing values to the Server) and detect false command attacks (sending malicious actuation commands to a PD). Our non-intrusive solution takes advantage of the existing PLCs in the CPS so that it can be easily integrated into current deployments at minimal cost.

Path Redundancy enables individual controllers to be able to collect object values on physical devices which are originally controlled by other controllers. However, it cannot automatically switch the control to other controllers when one is compromised. To solve this problem, we introduce a new CPS architecture in Section 5. The solution transfers the CPS from a tree structure to a fat-tree structure where each controller is connected to all the sensors and actuators. We introduce virtualization to embedded controllers to minimize the number of hardware devices needed. Our solution allows each sensor and actuator to

be simultaneously managed by multiple controller instances located on different physical controllers. One controller instance is designated as the Primary Controller Instance (PCI) and the others are designated as the Auxiliary Controller Instances (ACIs). The PCI is responsible for both sensing and actuation of the controlled device, while the ACIs act as the hot standby and are only used for data acquisition purpose. Once the PCI is been identified as anomalous, the server will isolate the original PCI from the network and designate an ACI as the new PCI. In addition, the server can isolate the ACIs who report invalid sensing data from the network. We discuss both passive and active switching strategy and emulate virtual controllers (called VPLCs) using Raspberry Pi devices and Docker. The emulation in Section 5 demonstrates that the emulated VPLCs are able to host $6 - 11$ controller instances simultaneously. Our solution overcomes the SPOFs of the current architecture and is applicable to various CPS networks with different communication interfaces, control frequency and application needs.

The rest of this dissertation is organized as follows. Section 2-5 provide detailed description for each defense mechanism. We present our conclusion and lay out a path for future work in Section 6.

2. SAFEGUARDING BUILDING AUTOMATION NETWORKS: THE-DRIVEN ANOMALY DETECTOR BASED ON TRAFFIC ANALYSIS *

2.1 Introduction

Building Automation Systems (BAS) employ distributed control networks to provide HVAC control, lighting control, as well as fire detection and security for an "intelligent building", and adopt BACnet as standard data communication protocol. BACnet employs different media types including MS/TP, LonTalk, ARCNET, and Ethernet. BACnet/IP is widely used because it effectively leverages IP internetworks to connect geographically scattered devices together [10]. However, the incorporation of IP protocols also creates a large attack surface for BAS, and hence providing a ripe target for potential cyber-attacks. To illustrate, Sochi Arena supporting 2014 Winter Olympics was found online and accessible without a password [11]. As of Sep. 2016, 18,358 BACnet devices and 51,252 SCADA devices (from Censys [12] and Shodan [13]) were exposed to the Internet. The Stuxnet attack [8, 14, 15] at Iran nuclear facility and Ukraine Power Outage [9] that took place on December 23rd, 2015 also show evidence of real threats to SCADA networks. In addition, most SCADA protocols, including BACnet, are designed without security measures such as authentication and encryption. Without sufficient security protections, attackers from the other side of the world are potentially able to access a BAS network, causing damage across long distance.

Effective protection of BAS networks against network attacks requires an in-depth understanding of real-world BACnet traffic. Due to data sensitivity and safety-criticality of BAS, network traffic measurement and analysis for BAS has not been widely carried out.

*Reprinted with permission from "Safeguarding Building Automation Networks: THE-Driven Anomaly Detector Based on Traffic Analysis" by Z. Zheng and A. L. N. Reddy, 2017. *26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, ©2017 IEEE.

Barbosa [16] observes that aggregated Modbus traffic is highly periodic with a low level of human interactions. In [17], Barbosa develops a periodicity based anomaly detector which considers traffic that only consists of strictly periodic bursts of packets. Wool [18, 19] developed a state-based anomaly detector for Modbus/TCP and Siemens S7 traffic based on the fixed sequence of commands sent from the operator workstation to PLCs. Previous anomaly detectors do not fit BACnet datasets because: (i) there is no fixed sequence of service arrivals in BACnet traffic; (ii) BACnet contains strictly periodic traffic as well as traffic that follows regular patterns and on/off models; and (iii) it also includes about 10% non-periodic traffic which makes anomaly detection difficult as this traffic will generate many false alarms for anomaly detectors based on the assumption of traffic periodicity.

We conduct in-depth BACnet traffic analysis from aggregated level and individual flow-service level. Our analysis reveals that BACnet contains three categories of traffic which are termed as "THE-Driven": Time-driven, Human-driven and Event-driven. Time-driven traffic consists of streams that follow periodic patterns, regular patterns and on/off models. Human-driven and event-driven traffic present non-periodic patterns and are observed to be about 10% of total traffic. Such behavior has not been observed in previous SCADA traffic analysis [16, 18–21]. Based on this behavior, we develop an advanced THE-Driven Anomaly Detector which adopts different mechanisms for each category of traffic. Our anomaly detector is shown to be able to effectively identify suspicious packets with high accuracy and low false alarm rate. Different SCADA protocols present different traffic behavior. In this section, we focus on understanding the BACnet traffic characteristics and develop anomaly detectors for such networks.

The goal of this work is to uncover and understand traffic behavior of real-world BAS network traffic, construct traffic models and develop an anomaly detector for BACnet traffic. We make the following contributions:

- Provide a detailed study of BACnet traffic collected over several days on a network with a number of BACnet devices and field panels.
- Show that BACnet traffic is made of not only strictly periodic traffic and human-driven traffic that have been found in other SCADA protocols, but also event-driven traffic as well as time-driven traffic that follow regular patterns and on/off models.
- Develop THE-Driven Anomaly Detector which employs interarrival-based detection for time-driven traffic, "safe range"-based detection for human-driven traffic, and volume-based detection for event-driven traffic.
- Construct flow-service level traffic models for time-driven traffic and validate the models using k-fold cross validation.
- Evaluate the anomaly detector using synthetic attacks and identify anomalies in our measured network.

The rest of the section is organized as follows. Section 2.2 provides an overview of BACnet protocol. Section 2.3 describes the threat model. Dataset and methodology are described in Section 2.4. Section 2.5 studies BACnet traffic at aggregated level: time series and packet size distributions. In Section 2.6, we provide a detailed analysis for each category of BACnet traffic and construct flow-service models for time-driven services. In Section 2.7, we present THE-Driven Anomaly Detector and evaluate it with synthetic attacks. We make comparisons with related work in Section 2.8. Finally, we conclude with a summary and suggestions for future work in Section 2.9.

2.2 BACnet Protocol Overview

BACnet/IP network architecture is shown in Fig. 2.1. As can be seen, it adopts a server-client architecture where the server is an *Operator Workstation* and clients are

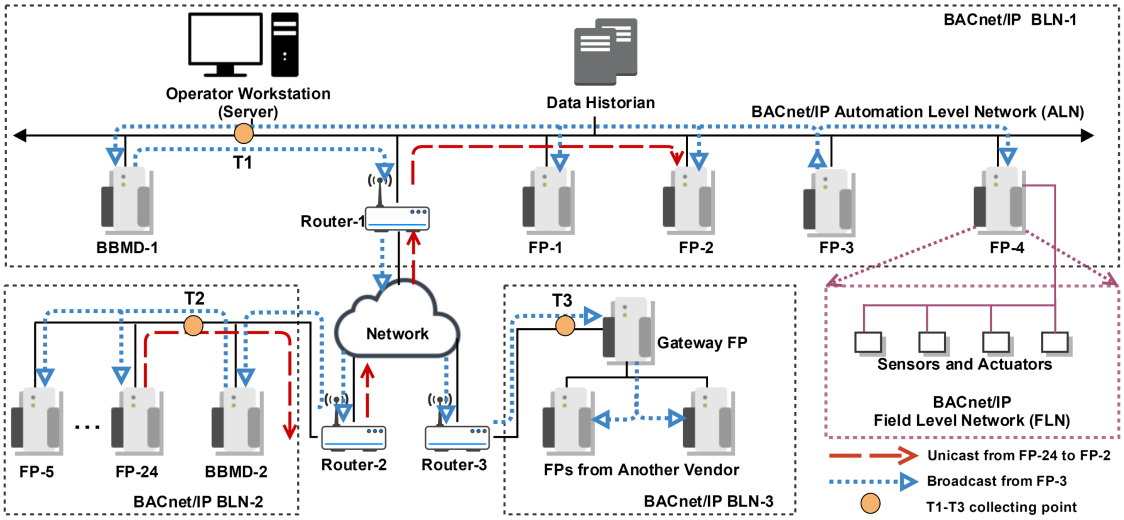


Figure 2.1: BACnet/IP network architecture and data collection points.

BACnet *field panels* (FPs). The server is connected to multiple clients via UDP/IP. *BACnet internetworks* are divided into smaller subnets which are often called *Building Level Networks* (BLN). A BLN comprises a bunch of geographically close BACnet/IP devices. Each BACnet FP may directly connect to several sensors or manage and manipulate some mechanical equipment which forms a *Field Level Network* (FLN). Normally proprietary protocols over serial links are used between FPs and mechanical equipment.

BACnet message types include *unicast*, *broadcast*, and *multicast*. Since broadcast is not allowed to pass across IP routers, a special device called *BACnet/IP Broadcast Management Device* (BBMD) is used to deliver broadcast across BLNs. The message is repackaged as a unicast and transmitted to a remote BBMD which then broadcasts on the remote subnet [22]. BBMD can be either a physically distinct device or integrated into a BACnet FP. Communication can take place between server-FP (BBMD), FP-FP, or

FP-BBMD. In Fig. 2.1, a unicast is sent from FP-24 to FP-2 and a broadcast is issued from FP-3.

BACnet defines a collection of abstract, network-visible view of data structures called *objects* to provide a basic model for data storage. The objects offer virtual data structures that enable accessing data without revealing internal designs of each device [23]. *Analog/binary input*, *analog/binary output*, and *device* are common objects. In addition, *services* provide functions to manipulate or access properties of objects. Those services are classified as *confirmed* or *unconfirmed*. BACnet adopts request-ack state machine for confirmed services. While unconfirmed services do not expect an ACK, another unconfirmed service may be sent as a response (*e.g.* WHOIS-IAM pair). A list of common BACnet services is shown in Table 2.1.

Abbr	Service	Type	Description
CAA	ConfirmedAcknowledgeAlarm	Confirmed	Tell the sender of an alarm that the alarm has been received
CCOV	ConfirmedCOVNotification	Confirmed	Tell the subscriber that a COV has happened in an object
CEN	ConfirmedEventNotification	Confirmed	Tell another device of an error or fault has occured
CPT	ConfirmedPrivateTransfer	Confirmed	Sends a vendor-proprietary message to another device
IAM	I-AM	Unconfirmed	Affirmative response to WHOIS, broadcast
IHAVE	I-HAVE	Unconfirmed	Affirmative response to WHOHAS, broadcast
RP	ReadProperty	Confirmed	Reads the value of a particular object property
RPM	ReadPropertyMultiple	Confirmed	Reads the values of multiple object properties
RR	ReadRange	Confirmed	Reads trend-longs of a BACnet device
SCOV	SubscribeCOV	Confirmed	Sent by a device to request that it be told of COVs in an object
TS	TimeSynchronization	Unconfirmed	Notify the device of the correct current time
WHOHAS	WHO-HAS	Unconfirmed	Ask which BACnet device holds a particular object, broadcast
WHOIS	WHO-IS	Unconfirmed	Ask about the presence of a particular object, broadcast
WP	WriteProperty	Confirmed	Writes a value to an object property
WPM	WritePropertyMultiple	Confirmed	Writes multiple values to different object properties

Table 2.1: BACnet service abbreviations and descriptions.

We define *aggregated traffic* as the whole traffic dump that we collected from a location. It contains different kinds of service packets over different host pairs. *Service stream* refers to a particular service request stream over different host pairs. We further define

flow as a unidirectional traffic stream from host A to host B, and *flow-service stream* as a specific BACnet service traffic extracted from a unidirectional flow.

2.3 Threat Model

BACnet is designed without security measures such as authentication and encryption. Most implementations do not check the source IP address or source device ID of a packet. When BAS networks are exposed to the Internet, attackers can send arbitrary packets to a BACnet device from outside the network.

We consider a BAS network that can prevent unauthorized BACnet packets from outside network and enforce source IP address checks. Such security protection is insufficient to prevent cyber-attacks from happening. We assume attackers can send packets to the BAS: either physically connect their machines to the network or leverage compromised devices to launch insider attacks. We aim to develop a BACnet-specific anomaly detector and consider the following attack categories.

1. **IP Spoofing and Data Injection.** Due to the lack of authentication, attackers can easily get around source IP checks by spoofing the IP address of an authorized BACnet device. IP spoofing is usually used to inject commands or data into an existing data stream. We assume the attacker can inject false commands to BACnet devices.

2. **Session Hijacking.** Attackers can potentially take over a session taking place between two victim BACnet devices. The attacker can interpose as one of the devices through IP spoofing and invoke ID (sequence number) guessing. Since BACnet devices employ a 1-byte sequence ID, it can be easily guessed or sniffed from the ongoing traffic.

3. **Reconnaissance Attacks.** Such attacks are used to gain valuable information about potential targets and usually come before other attacks. Similar to port scanning,

attackers are interested in finding what hosts are online, what services are running, and what firmware or operating system a host is using. For example, a WHOIS request can be sent to acquire IP address of one or multiple BACnet devices on the network.

4. **Denial of Service Attacks.** Attackers could stage *DoS attacks* by sending a high volume of BACnet or other packets to target devices. Moreover, consider a case where the server sends periodic packets to a FP through a router which also forwards traffic among other hosts. Attackers can launch *low-rate DoS attack* which periodically jams the router buffer, delaying or dropping the normal packets to the target device. Such attack is similar to the attacks described in [24].

5. **Buffer Overflow Attacks.** Such attack usually sends a single packet with an illegal packet size, trying to exploit insufficient bounds checks in the protocol implementation. Since BACnet does not regulate the size of different types of services, BACnet bounds checks haven't been implemented in existing IDS.

6. **Safety-critical Attacks.** WP and WPM are safety critical services as they are able to make direct changes to the system. Depending on different physical applications, a single write packet may lead the system into a critical state or cause significant damage to the entire network.

2.4 Datasets and Methodology

A realistic anomaly detector requires a comprehensive understanding of real-world BACnet traffic behavior. We worked with a utility company to collect traffic profiles in a BAS network from a university campus. Since all BACnet traffic in the network is BACnet/IP, we focus our study on BACnet/IP traffic.

2.4.1 BACnet Dataset

The BAS network in our study is illustrated in Fig. 2.1. The server, resides in BLN-1, administers 175 FPs located in about 100 buildings. FPs that are in the same building form a BLN (subnet). Depending on the building size and applications, each BLN consists of different number of FPs. More than 97% FPs in our network are Siemens FPs and the remaining FPs are from another vendor. BLN-2 includes 20 Siemens FPs and a BBMD. These FPs manipulate a number of application-specific physical controllers such as air handlers and chilled/hot water pumps. At FLN level, FPs use Siemens proprietary protocols over serial links to communicate with physical controllers. BLN-3 includes two third-party FPs and a Siemens *gateway FP* which passes all messages to/from third-party FPs. The gateway FP translates services and object references between the two manufacturers' equipment.

	Trace 1	**Trace 2**	**Trace 3**
Start Time	2014.10.28 10:26:45	2014.12.05 16:46:04	2014.10.28 14:12:56
Duration (days)	7	5	8
Capture Location	The Server	BBMD	Third-party FP
Flow Types	(i) Server-FPs(BBMDs) (ii) broadcast	(i) Server-FPs (ii) FP-BBMD	(i) Server-3rd PT FP (ii) broadcast
% Broadcast	7.59	3.56	14.01
Total # of Packets	31684226	6283156	2213111
Total # of Bytes	3173684038	877925687	171125677
Avg Pkt/sec	52.387	15.625	3.202
Avg Bytes/sec	5247.447	2183.256	247.579
Avg Pkt Size (bytes)	99	141	79

Table 2.2: Datasets description and statistics.

We used open-source Wireshark [25] to capture our traces with detailed information: timestamp, source and destination IP, port number, packet length, and data payload. We collected BACnet traffic at three vantage points for several days and obtained three traces: T1-T3. Statistics of T1-T3 are summarized in Table 2.2. T1 is collected at the server and

includes (i) traffic between the server and multiple FPs (or BBMDs); and (ii) broadcast traffic. T2 is collected between BBMD-2 and FPs in BLN-2. It includes (i) Server-FPs traffic and (ii) FP-BBMD traffic within BLN-2. T3 measures traffic at the third-party FPs. We choose these three vantage points because they can help us completely understand the traffic between all types of communication hosts: the server, FPs, BBMD, and third-party FP. T2 dataset is chosen because it contains FP-BBMD traffic in BLN-2. Some networks may contain FP-FP traffic as well. Since BACnet contains not only traffic between server and FPs, but also the FP-BBMD traffic and third-party traffic, only one collection at the server or a network choke point does not suffice.

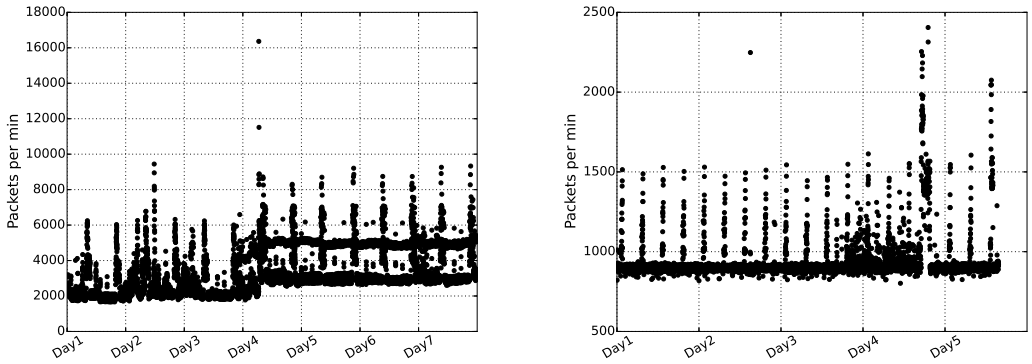
2.4.2 Methodology

Our analysis on BACnet/IP traffic is organized into two parts: aggregated level and flow-service level. First, we analyze aggregated traffic in terms of diurnal patterns, regularity from aggregated time series, and packet size distributions. Second, we break down aggregated traffic into individual flow-service streams and reveal traffic patterns of each stream. We classify BACnet services into three categories and develop flow-service traffic models for time-driven services. Such models are combined with other mechanisms to construct an anomaly detector for BAS networks. Techniques that are used in our traffic analysis include:

Autocorrelation and **FFT** are used to study periodicity of traffic from time domain and frequency domain, respectively. **Wavelets** present a signal's regularity from time-frequency domain, i.e., it decomposes a signal into different frequency components over time. **Histogram** of inter-packet arrivals are used to calculate time interval distributions of two consecutive request packets within the same flow-service stream.

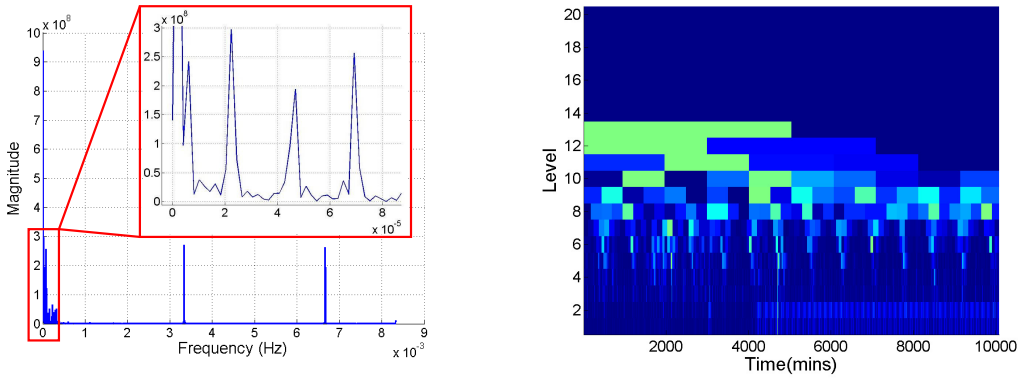
2.5 Aggregated BACnet/IP Traffic

2.5.1 Traffic Time Series



(a) **(b)**

Figure 2.2: (a) Aggregated T1 and (b) T2 time series do not present diurnal patterns.



(a) **(b)**

Figure 2.3: Aggregated T1 (a) FFT and (b) wavelets plots.

Fig. 2.2 depicts the plot of aggregated T1 and T2 traffic volume trends for the whole capture period. X-axis represents time in days and Y-axis denotes number of packets per

minute bin. It is observed that the traffic in all datasets **do not** present diurnal patterns. Observed traffic consists of steady volume of packets (T1: about 4,000 packets/min after Day 4; T2: about 900 packets/min) along with regular big spikes (T1: around 9,000 packets/min; T2: around 1,500 packets/min). Steady volume of traffic is contributed by regular time-driven services that are issued at small time scales (*e.g.* CPT and RPM every 10s to several minutes) as well as occasional non-periodic traffic. The non-periodic traffic is made up of human-driven and event-driven services. Regular big spikes are triggered by time-driven services with timers at large time scales (*e.g.* RR and RP polling trend-logs at every 12 hours in T1 and every 6 hours in T2).

We analyze the periodicity of aggregated traffic with autocorrelation/FFT/wavelets. Fig. 2.3(a) shows the FFT plot of T1 traffic. Spikes are observed at 3.33×10^{-3} HZ (300 seconds) and 6.67×10^{-3} HZ (150 seconds), which are contributed by regular time-driven services at small timers. We also observe a larger spike at 2.24×10^{-5} HZ (11.57 hours) which is caused by trendlog polling services (RR and RP). Fig. 2.3(b) shows that T1 traffic presents constant periodic behavior at level 8-10 (corresponds to roughly 12 hours). The traffic starts presenting periodicity at level 2 (smaller timers) since 4170th minute (Day 4), which is verified by the change of baseline in Day 4 shown in Fig. 2.2(a). Since 4170th minute, the baseline in Fig. 2.2(a) starts shifting between 3,000 and 5,000 packets/min. That behavior was caused by a policy change at the server (the server started issuing WHOIS messages to some FPs every 3-5 minutes).

In addition, a number of unusual spikes are observed in our dataset (see Fig. 2.2). Unusual big spikes are normally triggered by human-driven and event-driven services, or caused by significant changes in the system. Specifically, the largest spike at 4713th minute in Day 4 of T1 traffic is contributed by WHOIS and IAM messages. This is caused by a network scale power outage which leads to the disconnection and reconnection of all BACnet devices in the network. When the server sees a device failure, it will send WHOIS

messages to try to recover the failed device. Another large spike at 2010th minute in Day 2 was caused by a Change-of-Value (COV) event, which was verified in T1-CCOV traffic in Fig. 2.6(b). Similarly, the unusual spikes in T2 were mostly contributed by WHOIS and IAM messages (caused by device failures or configuration changes). These events were verified by system logs and conversation with network operators. Traffic trends of T3 are similar to T1 and thus are omitted due to space limitation. We conclude that aggregated BACnet traffic presents some regular behavior, but is not strictly periodic.

2.5.2 Packet Size Distributions

Statistical analysis is conducted to study BACnet packet size distributions. Dominant portion of the traffic, 96.2% of packets, are less than 200 bytes in size. This is considerably different from the observed packet sizes in regular IP traffic [26]. We also study packet size distributions over different service streams and message types (Req or Ack). Results in Table 2.3 show that 8 groups have unique packet sizes and 9 groups have $2-5$ packet sizes. Groups such as CPTReq and RRAck have multiple sizes of packets. Part of the reason stems from the fact that CPTReq includes different number of objects, and RRAck contains varied-length trend-logs. Overall BACnet traffic follows tight packet size distributions.

No. of sizes	Services (Bytes)
1	CAAAck (60), CAAReq (97), CCOVAck (51) CENAck (51), SCOVAck (60), TS (60) WPAck (60), WPMAck (60)
2 - 5	CCOVReq, IAM, IHAVE, RPReq, RPMReq RRReq, SCOVReq, WHOHAS, WPReq
Multiple	CENReq, CPTAck, CPTReq, RPAck RPMAck, RRAck, WHOIS, WPMReq

Table 2.3: Packet sizes for different BACnet services.

2.6 BACnet Traffic Analysis

2.6.1 BACnet Traffic Classification

We conduct an in-depth study at flow-service level to understand traffic patterns. Traffic pattern of any individual flow-service stream depends only on its end hosts and is independent of traffic behavior of other flows or services. Hence we break down each trace into a number of flow-service streams by filtering our dataset with the tuple <src IP, dst IP, service ID>. This significantly reduces the dimensional space and thus facilitates the modeling process. We focus on the request packets here. Based on how different services are generated, we summarize BACnet traffic into three traffic categories (called "THE-driven"): *Time-driven*, *Human-driven* and *Event-driven*. Time-driven traffic is normally generated by scheduled control programs that trigger service requests according to different timers. Such traffic presents time regularity and is not affected by real-time events of the network. Human-driven traffic includes requests that are directly generated by humans or through control programs (like WPM requests). Event-driven traffic includes service requests that are not generated by timers or humans. It depends on a broad range of events such as Change-of-Value (COV) on objects, system status change (normal/offnormal/fault), device failures, receiving alarms and other service messages.

Next, we classify BACnet services into the above three categories. First, services that present time regularity at various time scales are considered as time-driven traffic. We employ autocorrelation/FFT/wavelets and develop a module called *Time Regularity Classifier (TRC)* to automatically determine time regularity behavior of each service, which is further discussed in Section 2.6.3; second, we correlate services with human actions recorded in system logs and obtain a list of human-driven services; finally, services that are generated under different event cases are considered as event-driven, which is determined based on service usage cases specified in BACnet standard. Time-driven services are automatically

determined and the rest two categories can also be easily determined, as explained below. Table 2.4 presents the relationship between BACnet services and "THE-driven" category.

Note that this is not a mutually exclusive classification as a service can be classified into more than one category. For example, WHOIS is commonly used as event-driven: whenever a device is unaware of the interested device's IP address. This can be caused by various events such as connection failures, configuration errors, and faulty wiring. However, we do observe that the majority of WHOIS messages in our traffic present time-regularity and a few WHOIS messages are confirmed to be initiated by human operators from the workstation. Therefore we classify WHOIS service as the combination of three categories.

Similarly, RP is also observed in all three categories. We separate RP messages into *RP-trendlog* and *RP-others* according to object types (see Table 2.4). RP-trendlog is used together with RR to collect trend logs from FPs. They follow on/off models at large time scales. RP-others messages have much smaller traffic volume and are triggered by human (confirmed with system logs) or events (*e.g.* when receiving a CCOV message).

Categories	**Services**	**T1**	**T2**	**T3**
Time	CPT, RPM, RR, RP-trendlog, TS	43.52%	91.92%	34.25%
Human	WP, SCOV	0.38%	0.06%	0.64%
Event	CEN, CCOV	4.16%	2.02%	10.48%
Human+Event	RP-others	4.54%	1.49%	0.06%
Time+Human + Event	WHOIS, IAM, WHOHAS, IHAVE, WPM	47.40%	4.51%	54.57%

Table 2.4: Relationship between categories and BACnet services.

In Table 2.4, services that belong to multiple categories are listed separately. The specific percentage varies across our datasets, but the mapping between services and cat-

egories stays consistent. Compared with T1 and T3, T2 has higher percentage of time-driven services and less percentage of time + human + event services. That is because server-FP traffic (T1 and T3) has more WHOIS (T1: 33.8%, T2: 2.44%, T3: 29.12%) and IAM (T1: 11.9%, T2: 1.85%, T3: 11.89%) messages than T2 traffic (mostly FP-BBMD traffic).

We further identify the percentage of each category traffic in different services and conclude that overall BACnet traffic consists of about 90% automatically time-driven traffic, 5% human-driven, and 5% event-driven traffic. Human-driven and event-driven traffic are non-periodic. Note that the percentage of non-periodic traffic could be different with other vendors and system configurations. Our findings indicate control network traffic behavior is more varied than previous studies have indicated [16, 18, 19, 21, 27].

2.6.2 Anomaly Detection Strategy

The challenge of anomaly detection in BAS networks comes from the 10% non-periodic traffic. Using periodic model-based anomaly detector in [17] will result in many false positives. Moreover, the detector in [17] is not suitable for time-driven traffic that follows regular patterns and on/off models.

Therefore, we develop "THE-Driven" Anomaly Detector that employs different mechanisms for each category of traffic. Fig. 2.4 shows the architecture of our anomaly detector. It contains three stages: *data pre-processing*, *traffic classification*, and *detection*. In data pre-processing stage, *stream extraction* & *size checks* module divides aggregated traffic into flow-service streams and conducts packet size checks (further described in Section 2.7.1); next, we conduct time-regularity analysis at flow-service level to construct traffic models of time-driven traffic, and determine human-driven and event-driven services; finally, in detection stage, we employ different types of detectors for anomaly detection in different categories of traffic, as explained below. Services that belong to multiple cate-

gories are processed by multiple detection modules. Below is the detailed analysis of each category of traffic.

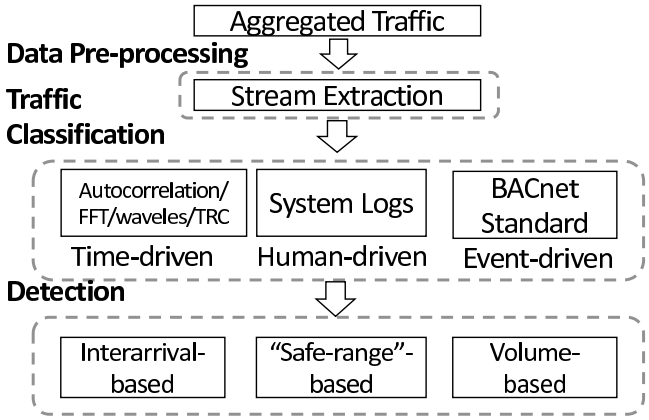


Figure 2.4: Architecture of THE-Driven Anomaly Detector.

2.6.3 Time-driven Traffic

The goal of time-driven traffic analysis is to determine if a flow-service stream presents time regularity behavior at different time scales, and what regularity patterns it follows.

First, we extract request timestamps and request interarrivals (i.e. time intervals between two consecutive requests) from each flow-service stream. Next, we run autocorrelation/FFT/wavelets on request timestamps to understand time regularity. Autocorrelation analysis seeks the time shift at which autocorrelation output reaches the maximum. Such time shift corresponds to the time periodicity. Similarly, FFT analyzes the periodicity from the frequency domain and wavelets can be employed to observe changes of periodicity over time. Furthermore, we develop *Time Regularity Classifier (TRC)* to automatically estimate traffic models and associated parameters. Specifically, for each sample of interarrivals, the module splits the sample into a sequence of bins of size k ($k \in [1, 50]$). Continuous k interarrivals are grouped in the same bin. The module takes each bin as a

unit and calculates statistical distributions of the sample under different bin sizes. Next, It chooses the bin size with the smallest standard deviation σ_{min}. If σ_{min} is too large or the number of interarrivals is too small, the module does not consider such samples as time-driven and filters them out. For the remaining samples, the module automatically determines the regularity behavior based on parameters such as *period* (validated with autocorrelation/FFT/wavelets results) and *packet counts per period* (equals to the chosen bin size). Regularity behavior of time-driven traffic is characterized as the following three types:

- **Periodic patterns** flow-services issue service requests regularly and periodically with fixed intervals. Such streams contain only one packet per period. This behavior can be characterized using a *period* parameter. Fig. 2.5(a) shows the autocorrelation plot of a periodic traffic. Periodic spikes are observed every 60 seconds.

- **Regular patterns** describes such time-driven flow-services that even though individual interarrivals do not look similar, periodic behavior is observed at larger time scales or over multiple packet arrivals. Regular patterns services contain multiple packets per period. We use *period* and *packet count* n parameters to describe such pattern. Fig. 2.5(b) shows the autocorrelation plot of a regular traffic. In addition to constant spikes every 60 seconds, it also shows some smaller spikes which are triggered by individual packets. This traffic consists of 11 packets arriving every 60 seconds.

- **On/off models** describes flow-services that follow on/off patterns. Multiple service requests are issued frequently during the on-time period, and the service sleeps when off-time starts. A fixed value is observed in both on-time and off-time of each flow-service stream. Such services present regularity at several hours time scale and the long off-time differentiates on/off models from regular patterns. Hence, besides

period and *packet count* n, we also employ *on-time* and *off-time* for characterization. Fig. 2.5(c) illustrates the autocorrelation plot of an on/off model traffic. Spikes are observed about every 12 hours (4.2×10^4 seconds).

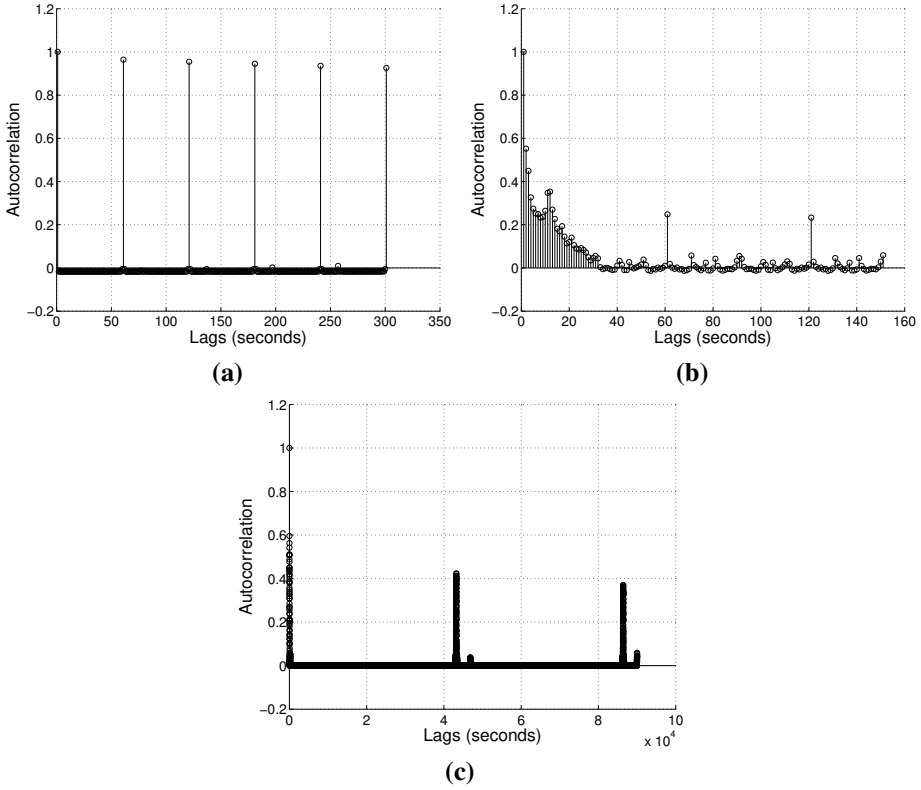


Figure 2.5: Autocorrelation plots of (a) periodic patterns; (b) regular patterns; and (c) on/off models.

These classifications are made automatically based on autocorrelation/FFT/wavelets analysis and TRC. This technique can be employed in different network settings with different deployment practices.

2.6.4 Human-driven Traffic

Human-driven traffic is generated by operators from the server or workstation. It constitutes about 5% of total BACnet traffic and does not present time regularity. System logs keep track of all human-driven packets which include: RP-others, WP, WPM, SCOV, WHOIS and WHOHAS. The most safety-critical human-driven services are WP and WPM because they are capable of making direct changes to the system. WPM is functionally equivalent to sending multiple WP messages within one operation. It is used to write a group of related values in one or more objects.

Within a week period of T1 traffic, there are 207 WP messages (sending to 91 unique objects within 29 unique FPs) and 233 WPM messages (sending to "schedule" object within 8 unique FPs). We confirmed that all these messages were initiated by operators. In addition to write packets, system operators can also read object values on FPs (through RP-others messages), enumerate devices and objects (through WHOIS and WHOHAS), and subscribe COV events (through SCOV messages).

2.6.5 Event-driven Traffic

Event-driven traffic is triggered by other service messages or changes in the system. Similar to human-driven traffic, it also does not present regular/periodic behavior and is small in traffic volume. It is observed that for a single event-driven service, many flows only have a few or even no messages. Therefore, rather than flow-service streams, we consider each event-driven service in *service streams* (streams of a particular service across flows). Event-driven services include CCOV, CEN, RP-others, WHOIS, and WHOHAS. Below is the description of these services.

CCOV is normally used in BAS to reduce the amount of network traffic spent on polling object values. It can be triggered by Change-of-Value (COV) events or human-driven Subsribe-COV (SCOV) messages. When an operator subscribes COV events on

objects within a FP or a physical device, the human interface software will generate SCOV messages to the objects. Once receiving the SCOV message, a CCOV packet will be immediately sent back to the subscriber. In addition, when the subscribed object value changes beyond the threshold (default in Siemens product is 1 unit on FPs and 0.25 unit on physical devices), a CCOV packet will be sent to the subscriber. Sometimes the subscriber further generates a RP-others packet to re-read the object values. Therefore, we classify CCOV and RP-others here as event-driven messages. Fig. 2.6 shows the time-series of SCOV, CCOV, and RP-others, where we can observe some correlations among those spikes.

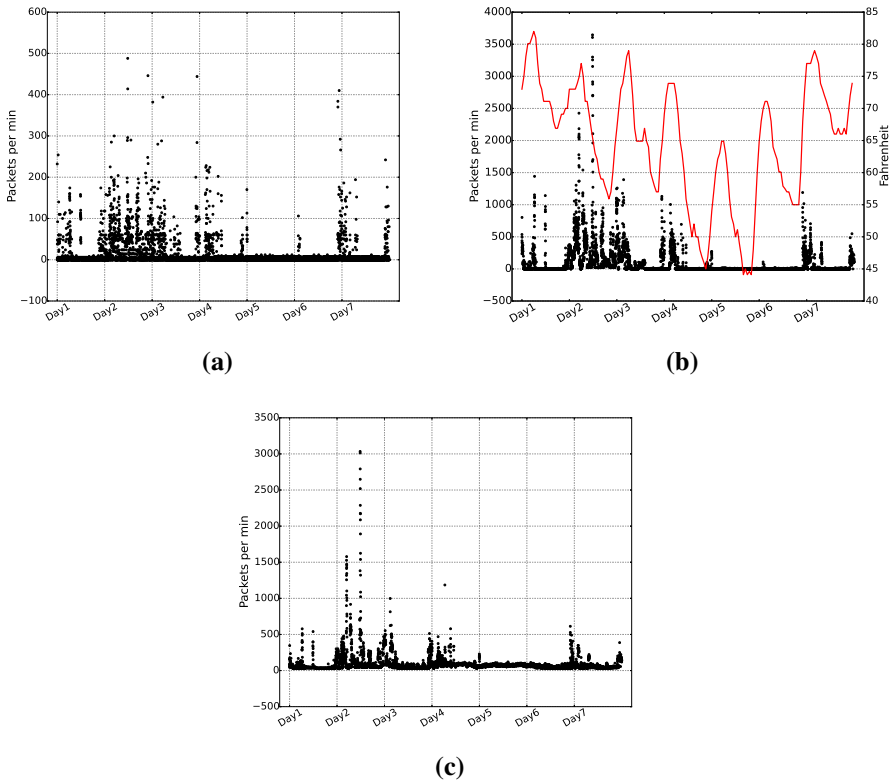


Figure 2.6: (a) T1-SCOV traffic; (b) correlation between T1-CCOV traffic and external temperature; (c) T1-RP-others traffic. Correlation of throughput spikes can be observed between SCOV, CCOV, and RP-others services. (bin = 1 min)

We further investigate physical meanings of the objects in CCOV and CEN packets. The physical objects vary from analog objects (such as room temperature, air/water pressure and flow) to binary objects (*e.g.* day/night mode). In addition, we compare the spikes of CCOV and CEN time series with external temperature changes and observe a certain degree of correlation, see Fig. 2.6(b). For example, the spikes at the beginning of Day3, Day 4 and Day 7 look related to the dramatic temperature changes. Because room temperature is only a small subset of objects in CCOV messages, not all correlations are apparent.

CEN messages convey alert notifications which may be triggered by device state (normal/ offnormal/ fault) changes or value changes. For example, if an alarm is set to room temperature at 78 degrees, an "out of range" alarm will be sent when the value is above or returns below 78 degrees. 85.7% alarms in our system logs are "out of range" alarms.

WHOIS and WHOHAS messages can also be event-driven when a BACnet device does not know the IP address of interested BACnet devices or interested objects. This can happen when a device gets disconnected or reconnected, due to a power outage incident. Some BAS networks do not send WHOIS and WHOHAS regularly as "ping" messages in order to reduce network traffic.

2.6.6 Flow Model for Time-driven Traffic

Based on time regularity observations of time-driven services at flow-service level, we construct traffic models for each flow, namely *Flow Model*. Non-periodic services are considered separately. Each flow is a combination of a few types of time-driven flow-service streams, which allows us to construct a Flow Model as a combination of several time-driven traffic patterns. Since the exact sequence of the arrival of different services is not always the same, we model each flow-service as an independent component and characterize its own time regularity behavior. For example, the traffic flow from the server to one of the FPs includes 5 time-driven services: RPM and TS follow periodic patterns, CPT

follows regular patterns, RR and RP-trendlog follow on/off models. We study all flows in our datasets and construct Flow Models. Flow Models and parameters are automatically learned using TRC.

Next, we construct *End Point Model* of a specific BACnet device. It can be thought of as a combination of several independent Flow Models. Each Flow Model represents a single flow traffic where this device serves as an end host. The Flow Model and End Point Model are applied to our THE-Driven Anomaly Detector to process time-driven traffic.

2.7 Anomaly Detection and Evaluation

This section describes how we can use THE-Driven Anomaly Detector to identify suspicious traffic. Section 2.7.2 – Section2.7.4 provide detailed description for modules in detection stage. Each module corresponds to one category of traffic. Even though these detection modules use different techniques, they share similar ideas which can be summarized as follows: 1. For each category of traffic, our anomaly detector combines statistical analysis with application needs to model different aspects of normal traffic: arrival behavior of time-driven traffic, object value range in human-driven packets, and volume of event-driven streams. 2. It automatically decides a reasonable threshold: $\mu \pm r \times \sigma$ $(r > 0)$ for time-driven, where μ is the mean and σ is the standard deviation of interarrivals, "safe range" for object values within human-driven packets, and $\mu + 3 \times \sigma$ for event-driven traffic volume. 3. We evaluate the detection performance with k-fold cross validation or our datasets. 4. We identify suspicious traffic in our measured network and explain the causes of such anomalies. In Section 2.7.5, we conduct synthetic attacks to show that our anomaly detector is effective in detecting various attacks.

2.7.1 Stream Extraction & Size Checks

Stream extraction & size checks module processes aggregated traffic and extracts timestamps and interarrivals of individual flow-service streams (for time-driven traffic)

as well as service streams (for event-driven traffic). This module also enforces packet size checks on services that have less than 5 distinct packet sizes (which covers 17 out of 25 service types) because malformed packets can have different packet sizes. The module raises alarms on packets with illegal packet sizes.

2.7.2 Interarrival-based Detection

2.7.2.1 Periodic/Regular Patterns

Interarrival-based detection module employs Flow Models (or End Point Models) to model **arrival behavior** of time-driven services. Since the timing of inter-packet arrivals does not precisely follow the timers, our model considers statistical distributions of model parameters to allow certain variances.

For each individual flow-service stream, the module learns traffic patterns and associated parameters. It fixes packet counts per period and calculates statistical distributions of other parameters (mean μ and standard deviation σ). For each parameter, an accepting window is created as $[\mu - r \times \sigma, \mu + r \times \sigma]$, where $r > 0$. The threshold is determined by r. Note that individual packets are considered as units for periodic patterns, whereas periods (contains n continuous packets) are considered as units for services that follow regular patterns and on/off models.

We conduct k-fold cross validation ($k = 10$) on our datasets to evaluate the correctness and accuracy of generated models. The module randomly partitions the original sample into k equally sized folds (1 fold for testing and $k - 1$ folds for training). The module constructs the traffic model based on the training data and tests the model with the testing data. It identifies suspicious units that falls outside the accepting window. If multiple packets arrive within the threshold, we consider the one which is closest to the mean as legitimate packet and raise alarms on other packets. This process is then repeated k times,

with each fold used exactly once as the testing data. The k alarm rates are averaged to produce a single estimation.

We run 10-fold cross validation with various thresholds ($r \in [1, 5]$). Average alarm rates with threshold $r \in [3, 5]$ are listed in Table 2.5. As is shown, the alarm rate decreases with larger threshold and it drops to about 0.5% when $r = 4$. The small alarm rate in our measured network validates our traffic models.

Threshold	Periodic	Regular
$r = 3$	0.894%	0.769%
$r = 4$	0.576%	0.503%
$r = 5$	0.317%	0.331%

Table 2.5: Alarm rates of periodic and regular traffic.

2.7.2.2 On/off Models

For services that follow on/off models, we consider each period as a unit and model the arrival behavior with four parameters of on/off models (see Section 2.6.3). The detection of on/off models is based on the assumption that suspicious periods are likely to violate more parameters than legitimate ones. For each period, we compute the distribution of four parameters and declare an anomalous period if the model parameters are different significantly from the observed distribution.

The module identifies 3-4 suspicious periods in most flow-service streams over the whole week (threshold $r = 3$). Table 2.6 presents some detected suspicious period IDs in T1 flow-service streams. Suspicious periods in each stream are ranked according to the number of bad parameters. It is observed that same suspicious periods are identified in different flow-service streams (*e.g.* Flow1-RP-trendlog and Flow1-RR both identify period #4, #3 and #7). Such periods will receive more attention and higher priority in

Flow ID	Service	# Periods	Suspicious Period ID
1	RP-trendlog	14	4, 3, 7
1	RR	14	4, 3, 7, 8
2	RP-trendlog	14	4, 5, 13
2	RR	14	4, 3, 8, 7
3	RP-trendlog	14	4, 5, 7, 8
3	RR	14	4, 5, 13, 8

Table 2.6: Detected suspicious on/off periods.

alarm reporting. We further confirmed with system operators that system level device reconfiguration happened in period #3 and #4, which validates our results.

A closer inspection of the above anomalies reveals that most suspicious packets/periods are caused by (i) system level or individual device failures; (ii) network noise and delays; (iii) event-driven or human-driven messages; and (iv) configuration and policy changes.

Interarrival-based detection can be deployed in front of a BLN or at a network choke point. The detection module inspects service combinations as well as individual flow-service patterns. Any network traffic that deviates from the traffic model is automatically blocked and reported as a security alarm. When a FP is newly introduced into the network, the Time Regularity Classifier (TRC) needs to collect at least 50 continuous interarrivals to generate Flow Models. Model generation time varies from a few minutes for periodic and regular services, to several hours (a complete period) for on/off models traffic.

2.7.3 "Safe range"-based Detection

For human-driven services, we employ "safe range" based detection to enforce acceptable **object values** in write packets. Since the most safety-critical human-driven services are WP and WPM, we focus on WP and WPM messages and limit an object value to be changeable within its safe range.

The safe range of different objects is determined based on physical meanings of these

objects and application needs. For instance, the room temperature setpoint can be limited to 65 to 85 degrees Fahrenheit, rather than 55 to 95 degrees Fahrenheit supported by some vendors.

Therefore, once the detector enforces safe range on objects, any write packets that tries to change an object value outside its safe range will be considered anomalous and rejected with an incident alert. This can potentially prevent the system from getting into a critical state, even if attackers have hacked into the network. System operators can determine safe range settings based on expert knowledge and application needs.

2.7.4 Volume-based Detection

Volume-based detection models **traffic volume** of event-driven service streams and identifies unusual spikes. Even though BACnet has its own alarm messages (*e.g.* CEN) and notification messages (*e.g.* CCOV), the system is easily flooded with a lot of alarms (we observed more than 110,000 alarms in a single day), which makes it difficult for human operators to identify the real problem.

The detection is based on the fact that an unusual change in event-driven throughput is caused by a rapid/widespread change in the network. Such changes may come from object values, policy and configurations, and network topology. Thus, compared with minor changes, big changes are more likely to be related to suspicious events and they deserve operators' attention.

We split each service stream into time bins and calculate mean μ and standard deviation σ of throughput distribution. Time bins with throughput larger than $\mu + 3\sigma$ are declared as anomalous. We test with different bin sizes (1-sec/1-min/5-min) and obtain similar results.

We run the module on all event-driven service streams. Fig. 2.7 shows results of several event-driven services with bin size of 1-min. It shows original traffic trends (top figure) and corresponding event incidents (bottom figure). The red horizontal line in top

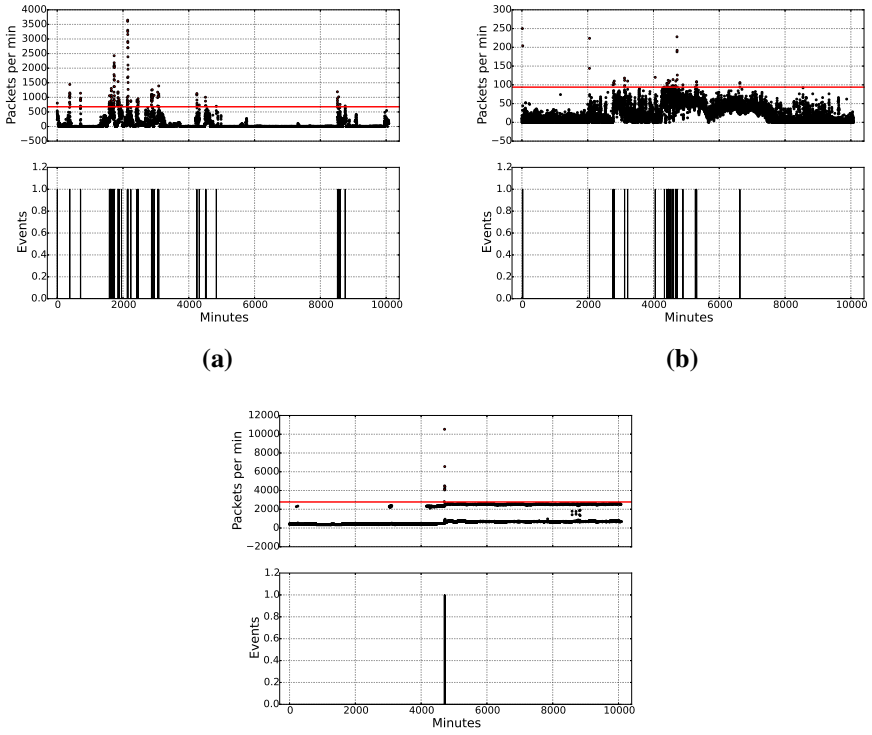


Figure 2.7: Volume-based detection results (bin = 1 min) (a) T1-CCOV traffic; (b) T1-CEN traffic; (c) T1-WHOIS traffic.

figures denotes threshold of anomalies. As is shown, the detector can effectively detect outliers in the traffic. It identifies 22 events in T1-CCOV, 16 events in T1-CEN, and 1 event in T1-WHOIS during the whole week. The spikes in CCOV (Fig. 2.7(a)) and CEN (Fig. 2.7(b)) are related to COV events, and spikes in T1-WHOIS (Fig. 2.7(c)) are caused by device failures. Results for other streams are omitted due to space limitation.

On average, the module identifies 1.51 alarms per service per day. Compared to current situation where system operators receive hundreds or thousands of alarms per service per day, this alarm rate is much lower and suspicious events can be effectively identified. Anomalies flagged by our detector were correlated to real events and incidents happening in the BAS.

2.7.5 Synthetic Attacks

We conduct several synthetic attacks to verify that our THE-Driven Anomaly Detector is effective in detecting unknown attacks described in Section 2.3. Our experiment assumes that the attacker is able to get access into a BAS network and get around network layer intrusion detection. We conduct the following attacks:

2.7.5.1 False Packet Injection Attacks.

Consider a case where the attacker spoofs the server's IP address and injects a number of time-driven service requests into the existing server-FP stream with random time intervals. The injected time intervals follow uniform distribution. We assume the attacker is not able to intercept normal packets. These injected packets are received together with legitimate packets and the victim is made to believe all packets are good packets originating from the server. However, time periodicity of such service requests are disturbed. We are interested to see if our interarrival-based detector can detect the attack and identify suspicious packets.

To simulate this attack, we use 9 folds data in Section 2.7.2 as training data and inject different percentage of attack packets into the remaining 1 fold (ratio of attack packets / normal packets $\in$ [0.1, 1]). Injected packets are labeled as "bad packets" and normal packets as labeled as "good packets". Next, we use interarrival-based detection to identify bad packets and compare the results with the labels.

Fig. 2.8(a) shows the ROC curve of periodic traffic detection with different percentage of attack packets. AUC is 0.988 when attack ratio is 1:10 and 0.887 when attack ratio is 1:1. When attack ratio is 1:1, TP rate reaches 1 when FP increases to 0.147. Note that this is the "detection rate of every bad packet", rather than the "detection rate of an attack". Since normal packets have some variances, it is possible that injected packets arrive closer to the mean of the distribution than normal packets so that false alarms are generated. Such

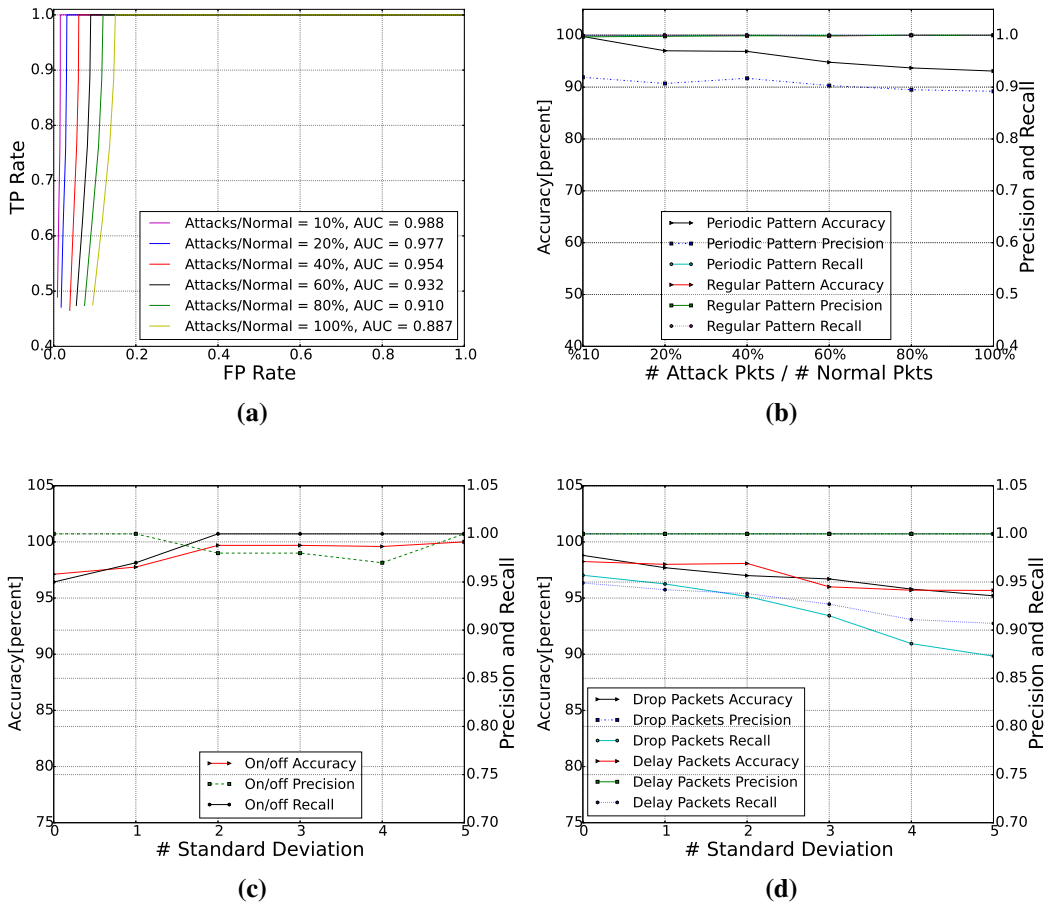


Figure 2.8: (a) ROC curve of injection attacks at periodic traffic ($r = 3$); results for (b) periodic and regular detection ($r = 3$); (c) on/off detection; (d) low-rate DoS.

cases result in a lower ROC curve with the increase of attack packets. Therefore, the actual detection rate of attacks is better than or equal to the results shown in Fig. 2.8(a).

Fig. 2.8(b) presents average accuracy, precision, and recall of periodic and regular detection using threshold as 3 standard deviations. Regular detection results are very good (above 99.7%) and better than periodic results. That is because regular detection identifies suspicious periods, rather than individual packets, and its detection rate is essentially the detection rate of attacks.

In addition, we inject attack packets into on/off models (attack ratio is 1:1). Fig. 2.8(c)

shows that average results for detecting on/off patterns injection attacks are optimal when the threshold is chosen between 2-4 standard deviations of the mean. As is shown, most metrics are above 97%. Therefore, we conclude that our interarrival-based detection can accurately detect injected packets with random time intervals.

2.7.5.2 Low-rate DoS Attacks.

Consider a case where the server sends time-driven packets to a FP passing through a network router. The router also forwards packets between the attacker and another host. Assume the attacker learns server-FP traffic patterns and launches low-rate DoS attacks at the router. The attacker performs *delaying attacks* and *dropping attacks* separately by changing the start time and duration of the low-rate DoS streams. Packets from the server will get delayed or dropped at the router.

We use NS-2 simulator to simulate this attack. We mix delayed/dropped packets (labeled as "bad packets") with normal packets (labeled as "good packets") and run our detector.

Fig. 2.8(d) shows detection results of (i) delayed packet attacks and (ii) dropped packet attacks with different thresholds. As is shown, when threshold is 3 standard deviations, accuracy is above 96%, precision is 1, and recall is above 0.91 under both attacks. In addition, such attacks are essentially similar to session hijacking attacks where the attacker issues packets with arbitrary time intervals.

2.7.5.3 DoS Attacks.

We launch synthetic DoS attacks in T1 traffic trace. Assume the attacker resides in the network and attacks the server with BACnet packets (100 bytes) over 900 Mbps bandwidth. Attacker generates more than 60M packets per minute. As the average packets received at the server is around 6,000 packets/min, volume-based detection can successfully identify such anomalies and send alarms on this attack.

2.7.5.4 *Reconnaissance Attacks.*

The attacker sends a WHOIS message to acquire IP addresses of other BACnet devices in the network. First, the WHOIS message is specified with all possible device IDs. It generates a large spike in IAM traffic trends and can be identified by volume-based detection; second, the attacker injects WHOIS messages to a FP without intercepting normal messages, this attack disrupts traffic patterns and is alarmed by interarrival-based detection.

2.7.5.5 *Buffer Overflow Attacks.*

The detector uses packet size checks to identify potential buffer overflow attacks. We send a number of malformed packets to a FP with illegal packet sizes. The detection rate is 100% for services that have a few distinct packet sizes (as described in Section 2.7.1).

2.7.5.6 *Safety-critical Attacks.*

We test "safe range" based detection using a number of WP requests with illegal object values. The packets get blocked and alarm accuracy is also 100%.

We conduct synthetic attacks for each type of traffic and summarize overall accuracy and false positive rates in Table 2.7. In addition, we measure the offline processing latency (i.e. time used for processing the full week samples in T1) of different mechanisms and present the average results in Table 2.7. We use a Ubuntu 15.10 system running on Intel Core i5-2320 3.00GHz CPU with 8G RAM to process the traces. Input data is the interarrivals of individual flow-service streams. Most modules take 30-60 ms to process interarrivals of streams, while on/off model takes 173.21 ms on average. Depending on the severity of the potential attacks, the checks can be carried out on each packet in real-time or with short enough delays since most building operations are not impacted by these small delays.

Table 2.7 also shows the online detection latency (i.e. the time between the arrival

	Avg. Accuracy	Avg. False Positive	Offline Processing Latency (ms)	Avg. Detection Latency
Periodic	95.87%	2.77%	27.38	real-time
Regular	99.31%	0.21%	56.94	1 period
On/off	98.77%	0.33%	173.21	1 period
Volume-based	100%	0%	31.67	1 time bin
Safe-range	100%	0%	32.44	real-time

Table 2.7: Detection results and processing latency ($r = 3$).

of an attack packet and the corresponding alarm) for different types of packets. Periodic and safe-range based detection can be processed in real-time because such services only contain one packet per period; whereas for regular and on/off models, the detector needs to wait a period time to detect attacks in the worst case. With volume-based detection, the detection latency can be 1 time bin at most. Since the fraction of on/off models traffic is less than 5% of the total traffic and the detection latency of other traffic is less than a few minutes, we conclude that our detector can successfully be used online.

In sum, THE-Driven Anomaly Detector can effectively identify suspicious traffic with small false alarm rate and low latency. It can be combined with network header field checks to provide better security protection. Since real-world network traffic is very sensitive and safety-critical, we are only able to collect traces from one site at this time. However, we believe other BACnet traffic follows similar behavior (three categories of traffic), despite the distribution of different services might vary. THE-Driven Anomaly Detector is able to automatically learn traffic behavior of different services. Therefore, our solution should be applicable to traffic from other sites or other SCADA protocols that present similar behavior. There is no pool of malware or attack incidents which can be used to draw real attack cases, so we evaluate our anomaly detector by injecting different synthetic attacks into the traces. This is the first work to observe and classify SCADA

traffic in this way, which is different from traditional IT traffic and pure periodic SCADA traffic. In future work, we would like to analyze BACnet traffic from other BAS networks and evaluate with real attack cases.

2.8 Related Work

2.8.1 SCADA Traffic Measurement

Earlier studies compared SCADA and BAS network traffic measurements with traditional IT traffic, plus verifying if models for the latter can be applied to SCADA traffic [16, 20]. It has been shown in [20] that diurnal patterns, together with self-similarity and heavy-tail distributions of connection sizes, are *not* present in Modbus and MMS traffic. However, our BACnet traffic dataset exhibits self-similarity, but not diurnal patterns or heavy-tail distributions. In [16], the authors analyzed Modbus traffic together with SNMP traffic from aspects of periodicity, throughput changes, and changes in number of hosts. The aggregated traffic is shown to exhibit high periodicity at 60s with a few exceptions (caused by human interactions). Authors in [21] use flow-based approach to collect BACnet traffic in a BAS network involved with a lot of human activity, and they *do* observe diurnal patterns. These works focus on traffic patterns at the network level while we provide detailed traffic analysis at higher flow-service level. Wool [18, 19] observed that Modbus/TCP and Siemens S7 traffic to and from a PLC is highly periodic and state-based: the HMI polls every PLC with a repeating sequence of commands that are issued at a fixed frequency, so alarms will be raised when a message appear out of its position in the normal sequence. Our study here has shown that the observed traffic in a BAS can exhibit considerably different behavior and the traffic behavior can vary across different service streams. Therefore, their state-based model does not fit our BACnet dataset.

2.8.2 IDS and Anomaly Detection for SCADA

Much research work has been focused on IDS and anomaly detection for SCADA networks [28–35]. Carcano et al. [36, 37] introduces a SCADA IDS solution based on critical state analysis and state proximity. Valdes and Cheung [38] propose a model-based intrusion detection for Modbus. It considers protocol header fields, communication patterns, and changes in service availability. They later evaluate two IDS: pattern-based detection for communication patterns among hosts, and flow-based detection for flow patterns [39, 40]. Wool et al. [18] also propose a model-based solution. Their models rely on the high periodicity of Modbus and Siemens S7 traffic as well as inter-packet relationship. Barbosa in [17, 27] proposed a flow whitelisting approach that enforces access control at the network level, assuming that most SCADA traffic is automated and repeated during training phase. Alarms are raised on flows with new IP addresses or port numbers observed in the detection phase. Barbosa [27] also propose an approach to learn timing cycles of Modbus and MMS at flow level, without digging into flow-service level. Such models work for highly periodic Modbus traffic but do not work well for MMS traffic with lots of non-periodic flows. In short, previous anomaly detectors do not work for BACnet because (i) BACnet has no fixed inter-packet relationship; (ii) it composes about 10% non-periodic traffic; and (iii) it requires more immediate per-packet anomaly detection than IP traffic.

2.9 Summary

Critical infrastructure networks have shown to be vulnerable to cyber attacks. This section analyzes real-world BACnet traffic in BAS networks and develops an anomaly detector considering different types of traffic in such networks. We conduct in-depth statistical analysis at aggregated level and individual flow-service level to understand traffic behavior at different types of end hosts. Our analysis reveals that (i) aggregated BAC-

net traffic does not exhibit diurnal patterns nor look strictly periodic because it consists of time-driven messages with different periodic behavior as well as non-periodic streams; and (ii) the non-periodic traffic includes human-driven and event-driven traffic.

Moreover, we construct Flow Models for time-driven traffic and validate them using k-fold cross validation. Our THE-Driven Anomaly Detector is shown to be able to effectively identify suspicious traffic in our datasets. We also evaluate the anomaly detector with synthetic attacks and show that it has accurate detection results.

3. SECURING CYBER-PHYSICAL SYSTEMS WITH ADAPTIVE COMMENSURATE RESPONSE*

3.1 Introduction

Many cyber-physical protocols and applications allow for situations where a single event can have a significant impact on the system, *e.g.* a single message from a controller may reset the temperature control of entire buildings. Similarly, a small burst of malicious actuation messages over a CAN bus in an automotive cruise control setting can cause the vehicle to operate at dangerous speed. In these examples, the adversaries leverage the inherent asymmetry between small effort and costly consequence to launch attacks that are either difficult to detect or difficult to counter in real time: the adversary can mount a small-footprint attack (*i.e.*, with a very small number of commands so that it cannot be discerned from nominal network traffic) that gives rise to difficult-to-detect anomalous behavior that in turn can cause a significant impact [8]. This renders the design of effective and non-intrusive countermeasures particularly difficult. In real-time scenarios, where timely delivery of messages is critical, the requirement to not unduly delay the delivery of messages (due to false positives or due to the need to collect authorization information) further complicates the design of protective countermeasures.

Research work in protecting CPS can be classified into two categories: communication networks approaches and system-theoretical approaches. The former category focus on the SCADA network and try to prevent malicious activities through methods such as IDS/anomaly detection [18, 28, 30, 41, 42], authentication [43–46], and encryption [47]. System-theoretical approaches, on the other hand, focus on the control system itself. Such

*Reprinted with permission from "Securing Cyber-Physical Systems with Adaptive Commensurate Response" by Z. Zheng, S. Jin, R. Bettati, and A. L. N. Reddy, 2017. *IEEE Conference on Communications and Network Security (CNS)*, October 2017, ©2017 IEEE.

protections are usually based on the inherent control plane dependencies and physical relationships. For example, dynamic watermarking [48–50] is effective to detect a number of stealty attacks including False Data Injection Attacks [51–54] and replay attacks [55].

In this section we present a framework, which we call Commensurate Response (CR), to harden CPS systems by addressing and reducing the asymmetry between the low cost of launching an attack using spurious and difficult-to-detect control messages and the costly effect that such an attack can have on the system. CR forces the footprint of the attack to be commensurate with its impact on the system. In some cases, this can be achieved by limiting the changes in actuation that can be triggered by a command. In order to initiate drastic changes in the actuation, CR would require the attacker to issue multiple, consistent requests. The larger the intended change, the larger the number of requests to be sent. This in turn increases the time for defensive measures, such as intrusion detection systems (IDSs), to detect the attack and to respond.

This section describes how to add CR capabilities to existing systems. We discuss where CR modules are to be added, and how they are to be designed and parameterized in order to protect the system without violating its original design requirements. We describe two types of CR designs, which we call *change-driven* CR and *criticality-driven* CR. Both designs provide commensurate response by dynamically controlling the system response latency as a function of current risk exposure and defense posture. In the case of change-driven CR, the risk exposure is measured in terms of the degree at which an issued command changes the plant operating setpoint. Given that an attacker may be masquerading as operator, large changes to the operating setpoint must be treated with suspicion, and thus the system response latency has to be temporarily increased to allow for defensive measures to react. For commands with a drastic change in setpoint, change-driven CR requires a longer response time for the plant to reach the setpoint, or the sender needs to issue multiple requests with smaller setpoint change. Instead of a single message to

increase the room temperature by 20 Celsius degree within one minute, attackers would now need to send 20 consistent messages or wait several minutes before reaching the setpoint. The additional footprints would greatly help the IDS to identify suspicious activities. Criticality-driven CR, on the other hand, monitors the system state and measures risk exposure as the current distance of the current operating point to the critical point, As the plant approaches a critical point, the CR module gradually tunes the system response to prevent the plant from reaching the critical point. In both cases, CR improves the system resilience and survivability [56] while guaranteeing the QoS (quality-of-service) and without affecting normal operations. We will describe how CR complements other defense mechanisms such as dynamic watermarking and deep packet inspection (DPI) to better secure CPS.

To evaluate our security mechanism, we provide a case study on automotive cruise control over a CAN bus and evaluate the performance of the two CR schemes. Results show that change-driven CR is effective when the attacker aims at manipulating the system setpoint (so-called setpoint attacks), and criticality-driven CR can be used against setpoint attacks and against situations when the attacker manipulates the system actuators (so-called actuation attacks). The benefits of our solution are the following. First, when the attacker tries to drive the system into a dangerous state, CR can effectively change the current state from unsafe to safe. Next, the CR module modifies system dynamics and expands acceptable operation range to accommodate different application needs. Finally, CR improves the system resilience and survivability and prevents stealthy attacks that cannot be detected by network-level protections (*e.g.* deep packet inspection).

We summarize the contribution of this section as follows:

- We propose Commensurate Response (CR) as a security mechanism that operates within the control domain to improve the security of a CPS while maintaining its op-

erational constraints. CR complements other security mechanisms, such as network-level IDSs.

- Change-driven CR brings IDS latency into system design consideration and enforces more effort for commands with larger setpoint change. It makes it difficult for attackers to cause drastic changes to a CPS.
- Criticality-driven CR tunes the system response as a function of how far the plant is from the critical point. This form of CR can be used against setpoint attacks and actuation attacks.
- We evaluate our solution using an automobile cruise control system operating over a CAN bus. We expose the system to attacks and demonstrate that CR improves the system resilience and attack survivability.

The rest of the section is organized as follows. Section 3.2 formulates the problem and introduces main components in a CPS. Section 3.3 discusses the attacker model. Section 3.4 provides a detailed system-theoretical study of both change-driven CR and criticality-driven CR. An evaluation with a case study is presented in Section 3.5. We make comparisons with related work in Section 3.6. Finally, we present our conclusions and lay out a path for future work in Section 3.7.

3.2 Problem Formulation

Fig. 3.1 depicts a typical CPS network architecture in an industrial control setting and its corresponding theoretical control model. As shown in Fig. 3.1(a), such a system may consist of a workstation, one or more PLCs (programmable logic controllers) or RTUs (remote terminal units), an IDS (intrusion detection system), and physical devices such as water pumps, valves, and temperature sensors. The PLCs and RTUs take control commands from the workstation and perform local sensing and actuation on the connected physical devices, in this case a pump and a valve. The following discussion applies to

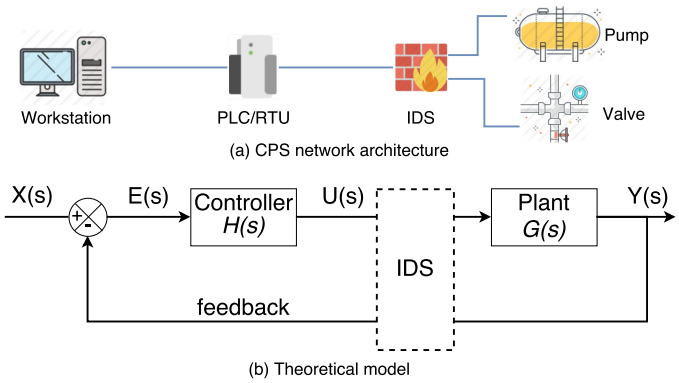


Figure 3.1: CPS network architecture and the theoretical model.

CPSs beyond traditional industrial control systems. For example, in Section 5 we will discuss the application of CR in an automotive cruise control setting.

The canonical model for such a control system contains three main components: a *physical plant*, a *controller*, and a *feedback loop*, see Fig. 3.1(b). The physical plant $G(s)$ refers to the system process. It takes one or more actuation signals $U(s)$ from the controller, and generates sensing values $Y(s)$ as output signals. The output signals are subsequently compared with the setpoint values $X(s)$ through the feedback loop to compute the error signal $E(s) = X(s) - Y(s)$, that is, the deviation of the output from the setpoint. The controller interprets $E(s)$ and adjusts the discrepancy by either attenuating or amplifying the input signal to the plant.

The communication interfaces between the controller and the plant can be of different forms: wired and wireless, Ethernet and serial. In order to protect the plant, IDSs and alert systems may be installed to monitor both the actuation signal $U(s)$ and the sensing signal $Y(s)$ on the communication paths between the controller and the plant. Alerts are issued whenever the plant approaches a *critical point*, that is, a state at which the operation of the system becomes either detrimental to the plant operation or even dangerous. Operating the plant at or above the critical point must be avoided.

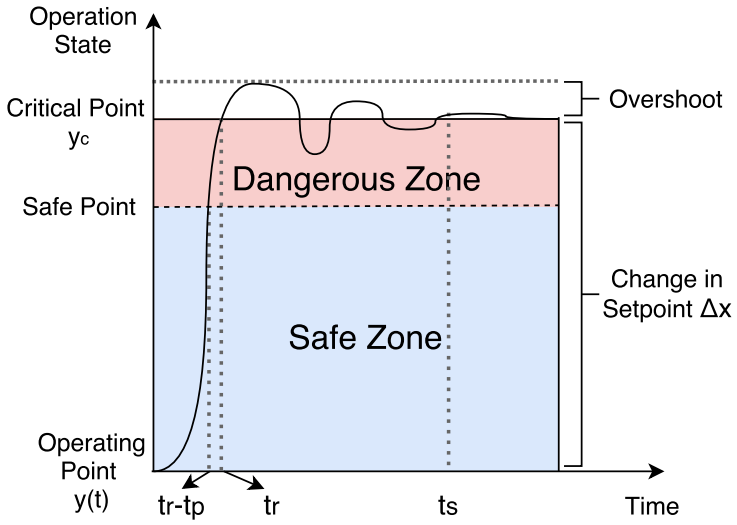


Figure 3.2: A sample plant step response $y(t)$.

In order to simplify the following discussion, we describe our approach using a simple system, consisting of a linear time-invariant (LTI) plant that is controlled by a PID controller. This simple system allows us to identify and focus on a small number of easily quantifiable objectives: system stability, rise time, overshoot, and settling time. This choice of system is not limiting: PID controllers are commonly used in a variety of settings, including temperature control, automotive cruise control, and the Boeing 747 autopilot control system [57]. CR can also be easily extended to more general systems, but parameters and objectives would become more cumbersome to describe.

Fig. 3.2 shows the plant step response diagram where the X-axis denotes time and the Y-axis denotes the state $y(t)$ of the system. Four major characteristics are used to describe the system response performance of PID controlled systems: (1) *rise time* t_r normally represents the amount of time the system takes to go from (0-10)% to (90-100)% of the targeted steady-state (we use the range of 1-99% in our study); (2) *settling time* t_s represents the time it takes for the system to converge to the targeted steady state; (3) *overshoot* denotes how much the peak level is higher than the targeted steady state, normalized

against the value of the latter; and (4) *steady-state error* denotes the difference between the actual steady state and the targeted steady state.

In addition, we define the *IDS response latency* t_p as the time for the IDS to detect an attack and to respond to it. The value for t_p is fixed for a given IDS. We note that for PID controllers in general the value for t_r is not dependent on Δx or the current operating point $y(t)$. Therefore we can say that $(t_r$ - $t_p)$ defines the largest safe operating point, that is, the operating point closest to the critical point at which the IDS can still detect and recover from an attack that intends to move the plant over the critical point. We call this point the *safe point*. The safe point divides the operation range, as shown in Fig. 3.2, into the *safe zone*, from where the IDS can recover from an attack, and the *dangerous zone*, where the IDS does not have sufficient time to recover before the plant reaches the critical point.

Consider a hypothetical case where a cruise control system with a PID controller increases the vehicle speed from $y(t) = 10$ mph to 100 mph (the critical point). If the rise time is $t_r = 1.5s$ and the IDS latency $t_p = 1s$, alerts will be issued at $(t_r - t_p) = 0.5s$, and the safe point is 85 mph. If the rise time is less than t_p, say $t_r = 0.5s$, then the IDS cannot recover from in time: When the IDS detects the attack, the vehicle speed has already exceeded 100 mph. Thus alerts should be issued at or before time (t_r - t_p). This requires the rise time t_r to be larger than or equal to t_p.

In many CPS networks, the requirement for IDSs to recover in a timely fashion is not considered when deciding the control parameters. In addition, given the plant and PID parameters, the rise time t_r is fixed and not dependent on the current operating point $y(t)$ or the setpoint change. Therefore, malicious control commands from the attacker can easily drive the system into a critical state within a short time, jeopardizing the safety and security of the system. The objective of CR is to actively manage the system response to maximize the safe operating range of the plant while at the same time maximizing the effort needed for an attacker to force the plant to or beyond its critical operating point.

Following textbook control theory, we represent a PID controller in the continuous-time domain as follows:

$$u(t) = K_p \cdot e(t) + K_i \int e(t)t + K_d \cdot \frac{e(t)}{t} \tag{3.1}$$

where $u(t)$ denotes the control signal to the plant, $e(t) = x(t) - y(t)$ the control error, $x(t)$ the setpoint, and $y(t)$ the measured output from the plant. The control parameters are proportional gain K_p, integral gain K_i and derivative gain K_d. In the Laplace domain, the transfer function of the PID controller $H(s)$ becomes:

$$H(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d \cdot s \tag{3.2}$$

where s is the complex frequency.

We derive the close-loop transfer function $\frac{Y(s)}{X(s)}$ of the overall system as:

$$Y(s) = G(s) \cdot U(s) \tag{3.3}$$

$$U(s) = H(s) \cdot E(s) = H(s) \cdot (X(s) - Y(s)) \tag{3.4}$$

$$\frac{Y(s)}{X(s)} = \frac{G(s) \cdot H(s)}{1 + G(s) \cdot H(s)} \tag{3.5}$$

In practice, the system designer chooses the PID parameters (K_p, K_i, and K_d) using one of several methods, such as Ziegler–Nichols [58] or Tyreus-Luyben [59]. Usually, the resulting PID parameters are improved through additional tuning to meet the following design constraints:

1. The rise time t_r is upper bounded by the system responsiveness requirement ($t_r \leq t_{max}$), especially for systems that consider responsiveness as a functional requirement (such as systems with hard deadlines);

2. Constraints on system *stability* (poles on the Left-Hand-Plane), *controllability* (rank issue), and *observability* (rank issue) need to be satisfied.
3. Application-specific requirements such as rise time, settling time, overshoot, and steady-state error need to be satisfied as well. Specific constraint values for these parameters vary across different applications and systems.

Response	Rise Time	Overshoot	Settling Time	S-S Error
K_p	Decrease	Increase	NT	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	NT	Decrease	Decrease	NT

Table 3.1: The effects of increasing each of Kp, Ki, Kd.

Table 3.1 illustrates how PID parameters related to system dynamics, and NT means not definite trend. [60] For example, in order to increase the rise time t_r, we can decrease K_p or K_i. However, the change in PID parameters will also result in the change in overshoot, settling time, and steady-state error. Similar tables are commonly used as guidelines for further PID tuning.

3.3 Attacker Model

Attackers who have gained access to a CPS may launch attacks from different locations in the control loop. We identify two types of attacks: setpoint attacks and actuation attacks. Fig. 3.3 demonstrates the attacker model considered in our work. We assume that attacks are perpetrated by insiders who either have direct access to the CPS network or who are capable of compromising devices in the CPS. They may attack the operator's workstation where the setpoint is issued, the controller, sensors, or communication channels. We do not consider attacks on sensors in this work and assume that they function correctly. Solutions for tolerating attacks on sensors have been previously proposed in [44, 45].

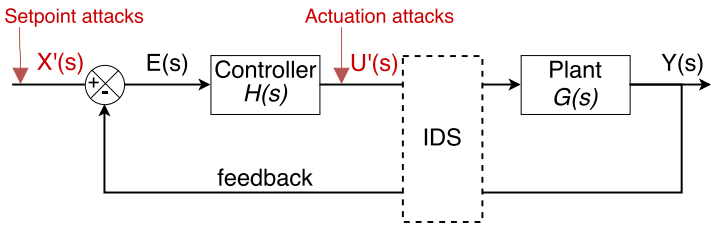


Figure 3.3: The attacker model considered in our work.

3.3.1 Setpoint Attacks

Attackers may inject malicious commands that trigger big changes to the setpoint values issued to the controller. The objective could be to change the setpoint to drive the system into a critical state. Attackers may achieve this through compromising the operator's workstation or the HMI (human-machine interface), or by hijacking the existing communication session between the workstation and the controller. If unchecked, the corrupted setpoint signal $X'(s)$ is then executed by the controller.

3.3.2 Actuation Attacks

Rather than changing the setpoint, an attacker may directly send malicious actuation commands to the plant on behalf of the controller, and so drive the system into a critical state. This attack can be achieved by (1) compromising the existing controller; by (2) hijacking the controller-plant communication channel and launching a man-in-the-middle attack; or by (3) having unauthorized devices masquerade as *bona fide* controllers and issue malicious actuation commands. We assume that the IDS monitoring points are judiciously placed, and malicious actuation commands cannot be inserted between the IDS and the actuators in the plant. Thus, the IDS monitors and processes any corrupted actuation signal $U'(s)$.

3.4 Adaptive Commensurate Response

In order for the IDS to respond in time, $t_r \geq t_p$ must hold. If this is not the case, the system designer has two options: replace the IDS by one with smaller detection latency and so decrease t_p; or change the plant dynamics to control the response time t_r. Often, intrusion detection in CPSs depends on correlation of actuation and sensing data over time, and so sufficiently reducing t_p may not always be feasible. We therefore choose to explore modification of t_r in the control plane. We are interested in controlling the system response under certain scenarios, so that the IDS and system operators have more time to detect suspicious packets when needed and as a result can take remedies accordingly.

Controlling the system response, and as a result modifying t_r, naturally controls how quickly a change of the plant state can take place as a result of the control input or of actuation commands. Attackers who attempt to circumvent CR have to send either more commands or commands with more aggressive setpoint or actuation requests. These actions will generate unusual spikes in the network traffic or control traffic with anomalous command sequences, both of which are easy to detect by the IDS.

3.4.1 Methodology

Fig. 3.4 shows the system model with Commensurate Response where the CR module $C(s)$ is inserted in the forward path. In practical cases, $C(s)$ can also be inserted in the reverse path, and the theoretical analysis remains the same. The CR module $C(s)$ can be designed to operate in two modes: *conservative mode* and *aggressive mode*.

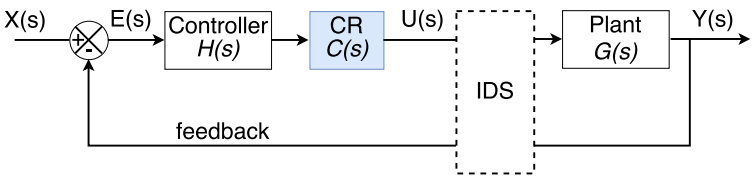


Figure 3.4: System model with adaptive Commensurate Response.

3.4.1.1 Conservative-Mode CR

CR in conservative mode modifies the original signal on the time axis by either adding a *time delay* or a *time scaling* factor the original signal. The time delay module postpones the signal by a constant time τ; and the time scaling module scales the signal on the time axis by a factor a. In discrete time, where signals are represented in the form of individual packets, the time scaling module can be implemented in two different ways: (1) fix packet count while increasing interpacket arrivals; or (2) fix interpacket arrivals while requiring more packets to be sent for a given action. We assume that the additional packets and latency have little impact on the bandwidth or network performance, and normal operations will not be affected.

3.4.1.2 Aggressive-Mode CR

When operating in aggressive mode, the CR module is designed so as to essentially replace the original controller with a new one that has the same structure but different control parameters. By selecting the parameters of the CR module the designer can control the system response while maintaing any desired system properties. We focus on this mode in the following discussion.

A PID controller can be represented as:

$$H(s) = \frac{K_p \cdot s + K_i + K_d \cdot s^2}{s} \tag{3.6}$$

We design $C(s)$ as:

$$C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2} \tag{3.7}$$

The new PID controller $H(s) \cdot C(s)$ turns out to be:

$$H(s) \cdot C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{s} \tag{3.8}$$

K_p^{cr}, K_i^{cr}, and K_d^{cr} become the new PID controller parameters. The parameters should be chosen properly to meet the design requirements discussed in Section 3.2. K_p^{cr}, K_i^{cr}, and K_d^{cr} now become functions of the *scheduling variable* $v(t)$:

$$K_p^{cr} = f_p(v(t)), \;\; K_i^{cr} = f_i(v(t)), \;\; K_d^{cr} = f_d(v(t)) \; .$$

We consider two different scheduling variables: (1) Δx: the change of the setpoint between two consecutive setpoint packets; and (2) $|y(t) - y_c|$: the distance between the current sensing value $y(t)$ and the system critical state y_c. CR that uses Δx is called "change-driven CR" because it relies on the degree of setpoint change. CR that uses $|y(t) - y_c|$ is called "criticality-driven CR" because it is based on the closeness of the current operating point to the critical point. Both CR strategies turn the control problem from linear to non-linear. CR module changes t_r from a fixed value to a function of $v(t)$. Instead of directly solving the non-linear problems, we employ *gain scheduling* to convert them into a set of linear problems, as discussed below.

3.4.2 Change-driven Commensurate Response

The idea of change-driven CR is to (1) make sure that $t_r \geq t_p$ and (2) increase t_r (slow down the system response) when a drastic change to the operating setpoint is about to take place. This mechanism not only provides enough time for the IDS to detect suspicious packets, but also enforces more effort when a significant change is about to take place in the system. Thus, attackers who aim to cause a big change to a system are forced to slow down by either waiting for a longer t_r or sending more setpoint commands, with smaller

setpoint changes. The additional response latency or messages will greatly facilitate the IDS to identify suspicious activities. In this case, we consider attackers may launch setpoint attacks, but not actuation attacks.

The transition of t_r from a static value to a function of Δx makes the size of dangerous zone to be dependent on Δx. As is shown in Fig. 3.5(a) and 3.5(b), change-driven CR is able to slow down the system response and extend the range of the safe zone while keeping the same time (t_p in this case) to reach the setpoint (the critical point here). Change-driven CR can change the current operating point from dangerous to safe and prevent drastic changes in a system. When a new setpoint arrives, change-driven CR will be triggered and the control parameters will be adjusted. The control parameters remain consistent as the plant approaches the setpoint.

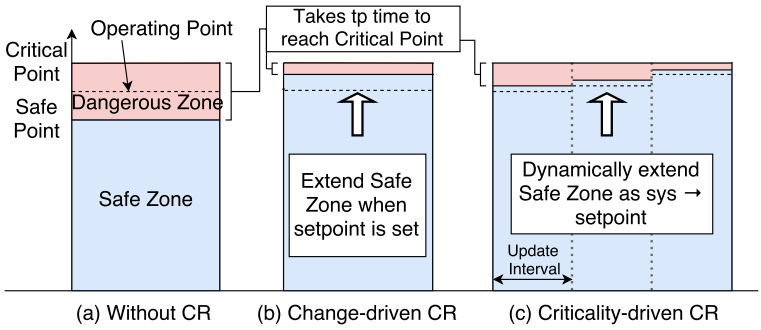


Figure 3.5: System effect of adaptive Commensurate Response.

Fig. 3.6 shows the theoretical model for change-driven CR. When a new setpoint command arrives, the absolute value of the setpoint change between two consecutive messages (Δx) is computed and used by the CR module to decide the new t_r. Given the dynamics of a plant, design steps for change-driven CR are as follows:

1) Starting from the original controller whose parameters (K_p, K_i, and K_d) are determined by the existing PID tuning algorithms, measure the original rise time t_r^0.

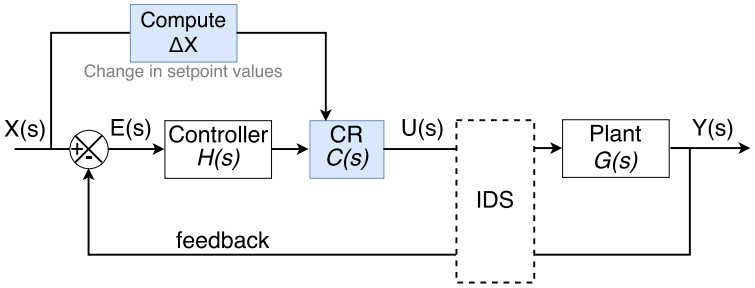


Figure 3.6: System model for change-driven CR.

2) Divide the largest possible setpoint range [0, Δx^{max}] into several intervals; design scheduling policy to match Δx^i (i^{th} interval of [0, Δx^{max}]) to a rise time t_r^i ($t_r^i \in [t_p, t_{max}]$). A larger Δx^i corresponds to a larger t_r^i.

3) For each t_r^i, obtain a set of PID parameter tuples that satisfy the design requirements.

4) Use greedy algorithm to select a PID parameter tuple for t_r^i. We select the one that has the smallest geometrical distance to the selected PID parameter tuple of $t_r^{(i-1)}$.

5) A scheduling table is created which projects Δx^i to a rise time t_r^i and further to a PID parameter tuple.

6) When a Δx arrives, the CR module looks up the scheduling table, locates the Δx interval, and chooses the PID parameters as K_p^{cr}, K_i^{cr}, and K_d^{cr}. $C(s)$ can be designed as $C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2}$ and inserted into the CPS.

Even though the CR module resides on the actuation path in the theoretical model, in practical cases, $C(s)$ can be applied to the actuation signal $U(s)$, or the sensing signal $Y(s)$, or both.

3.4.3 Criticality-driven Commensurate Response

In criticality-driven CR, the distance between the current operating point and the critical point, $|y(t) - y_c|$, is selected as the scheduling variable $v(t)$. Criticality-driven CR enforces state-based protection so that a *smaller* distance to the critical point results in

a *larger* response time t_r. Attackers who aim to sabotage the plant will find it difficult to drive the system into a critical state. Both setpoint and actuation attacks result in the change of $y(t)$, so criticality-driven CR (triggered by $|y(t) - y_c|$) is applicable to both attacks. Criticality-driven CR improves the system resilience and attack survivability from the system-theoretical perspective.

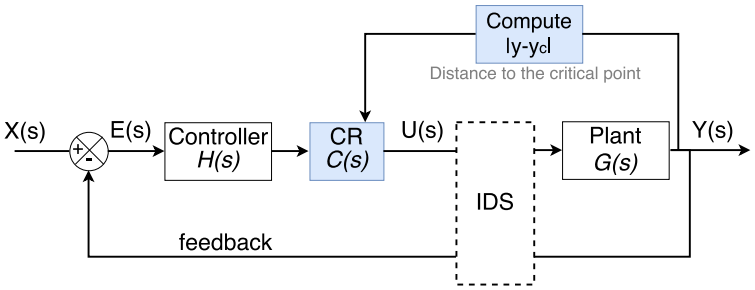


Figure 3.7: System model for criticality-driven CR.

Fig. 3.7 shows the system model for criticality-driven CR. Different from change-driven CR, criticality-driven CR increases the resistance of the system response as the plant gets closer to the critical point y_c. As is shown in Fig. 3.5(c), criticality-driven CR gradually slows down the system response (expands the safe zone) as the system approaches the critical point. Since criticality-driven CR relies on $y(t)$ to decide t_r, we assume that $y(t)$ is valid and cannot be modified by the attackers.

The size of the dangerous zone in Fig. 3.5(c) is continuously changing, but the time to reach the critical point, namely t_p in this case, remains constant. Criticality-driven CR essentially corresponds to the "two-second rule" in defensive driving. The two-second rule suggests a driver should ideally stay at least two seconds behind any vehicle that is directly in front of his or her vehicle, keeping a safe distance at any speed. The controlling variable is the vehicle speed and the dangerous zone is the safe distance to the car in front.

Criticality-driven CR gradually slows down the vehicle speed as it gets closer to the car in front, while maintaining the two-second rule. Our model provides the theory behind the two-second rule and can be applied to many other application scenarios. The design steps for criticality-driven CR are similar to the ones for change-driven CR except that state information is fed into the CR module and a *smaller* $|y(t) - y_c|$ corresponds to a *larger* t_r.

Compared to the existing practices where a fixed "safe-range" is enforced at the firewalls or the IDS, our solution aids in the choice of "safe-range" value and makes the choice of safe point more dynamic and flexible to accommodate different application and operating scenarios. Both change-driven CR and criticality-driven CR can either be applied separately or at the same time in a CPS, which is further discussed in Section 3.5.

3.5 Case Study: Automobile Cruise Control over a CAN Bus

3.5.1 System Model & Design Requirements

3.5.1.1 System Model

Automotive cruise control tries to maintain a constant vehicle speed despite external disturbances, such as wind and road grade. It employs a PID controller [61] to automatically control the speed of a vehicle. It is a typical intermediate-value control problem [62].

Fig. 3.8(a) shows a simple automotive cruise control system [63] where messages are transmitted over a CAN (Controller Area Network) bus. The system contains a steering wheel, a cruise control ECU (Electronic Computing Unit), a throttle valve and speed sensors. The ECU controls the throttle valve through a vacuum actuator and takes sensing values from the sensors which monitor the vehicle speed and throttle position. Fig. 3.8(b) illustrates the theoretical model. A vehicle of mass m is acted on by a control force u adjusted by the throttle valve. The force u is dampened by a resistance force of bv in the opposite direction. The resistance force bv represents disturbances such as the rolling resistance and the wind drag in the horizontal direction. We assume the control force u can

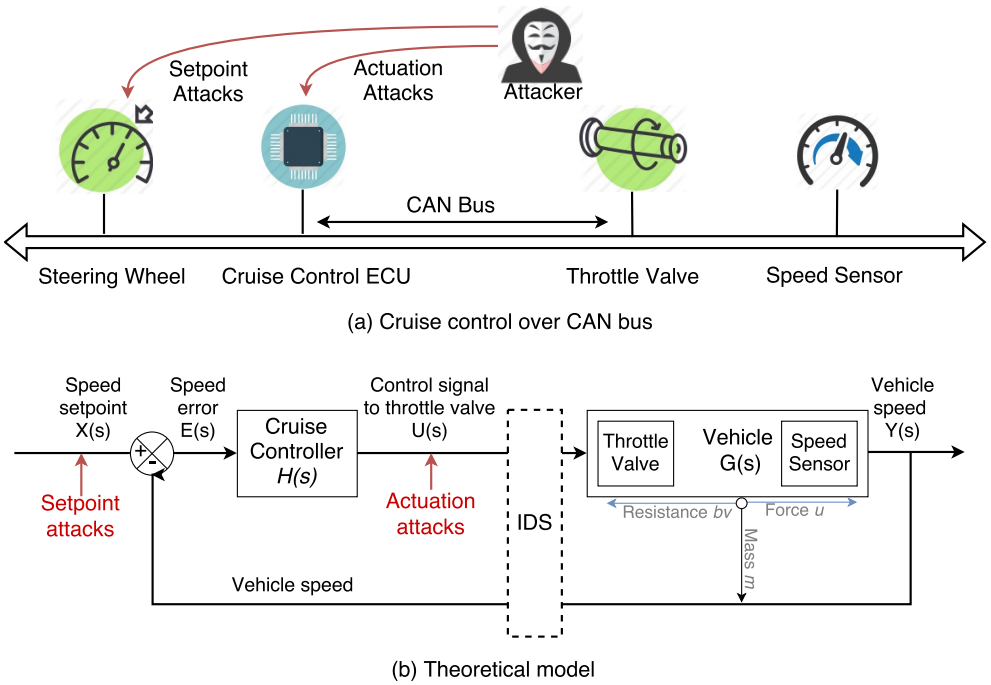


Figure 3.8: System diagram of automobile cruise control.

be directly controlled and thus we take u as the control input. We neglect the dynamics of powertrain and tires that generate such a force.

Our cruise control system can be modeled by a first-order mass-damper system: $m\dot{v} + bv = u$, where v is the vehicle speed, $\dot{v}$ is the acceleration, and b is damping coefficient. The output value y is chosen to be the speed of the vehicle: $y = v$.

The open-loop transfer function of the vehicle therefore is:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{ms + b} \qquad [\frac{m/s}{N}] \tag{3.9}$$

As an illustration, we choose the following parameters according to the example in [63]: vehicle mass (m) = 1000 kg; damping coefficient (b) = 50 N.s/m; nominal control force (u) = 500 N.

3.5.1.2 *Intrusion Detection Systems*

A number of IDSs have been developed for the CAN bus to monitor and detect suspicious traffic inside a vehicle. Cho and Shin [64] leveraged clock skew as a fingerprint of different ECUs to detect fabrication, suspension and masquerade attacks. Due to the statistical nature of the underlying detection mechanisms, the online detection latency can reach several seconds. In this example, we assume a detection latency t_p of 8.5 second.

3.5.1.3 *Design Requirements*

In the open-loop step response, when the vehicle is given a step input of 500 Newtons force ($u = 500$), it will reach a maximum speed of 10 m/s (22 mph). The vehicle should be able to accelerate to that speed within $13s$ with a maximum overshoot of 10%. Moreover, the vehicle should not reach the setpoint within the IDS detection latency t_p. In short, the design requirements are the following: (1) t_{max} = $13s$; (2) overshoot $< 10\%$; (3) t_p = $8.5s$; and (4) $t_p \leq t_r \leq t_{max}$.

3.5.2 Original System without CR

We use Ziegler-Nichols method to choose the initial PID parameters as $K_p = 289.74$, $K_i = 30.86$, and $K_d = 0$. The original PID controller is:

$$H(s) = \frac{K_p \cdot s + K_i + K_d \cdot s^2}{s} = \frac{289.74s + 30.86}{s} \tag{3.10}$$

The close-loop transfer function for this cruise control system is:

$$\begin{aligned} \frac{Y(s)}{X(s)} &= \frac{G(s) \cdot H(s)}{1 + G(s) \cdot H(s)} \\ &= \frac{K_p \cdot s + K_i + K_d \cdot s^2}{(m + K_d) \cdot s^2 + (K_p + b) \cdot s + K_i} \\ &= \frac{289.74s + 30.86}{1000s^2 + 339.74s + 30.86} \end{aligned} \tag{3.11}$$

Fig. 3.9 depicts the system step response of the original PID controller where the rise time is $8.0s$. With a single malicious setpoint message the attacker can drastically increase the vehicle speed (*e.g.* from 10 to 100 mph) within $8.0s$. Thus, the IDS on the CAN bus fails to detect suspicious packets in time because the detection latency is $t_p = 8.5s$.

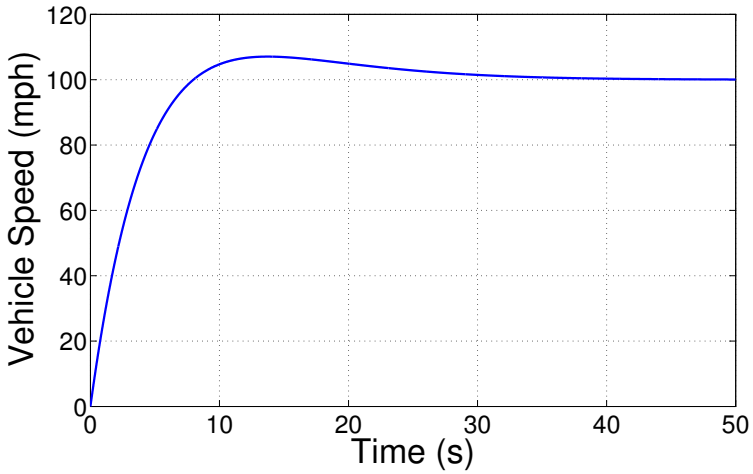


Figure 3.9: Original cruise control step response with rise time of $8.0s$.

3.5.3 System with Change-driven CR

Now, we insert change-driven CR module into the system. We adopt the aggressive mode CR where $C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{K_p \cdot s + K_i + K_d \cdot s^2}$. Thus, the new PID controller $H(s) \cdot C(s)$ turns out to be:

$$H(s) \cdot C(s) = \frac{K_p^{cr} \cdot s + K_i^{cr} + K_d^{cr} \cdot s^2}{s} \tag{3.12}$$

Given the plant dynamics $G(s)$, we run simulations to estimate the system response time t_r with different PID parameters and follow the design steps in Section 3.4.2 to create a scheduling policy, see Table 3.2. We consider 9 ranges for Δx, each of which corre-

sponds to a t_r and a PID parameter tuple. The critical point of the system is 100 mph. Any speed below 100 mph is considered safe for people and the vehicle. Decreasing K_p^{cr} and K_i^{cr} may increase t_r.

Δx (mph)	t_r	K_p^{cr}	K_i^{cr}	K_d^{cr}
0 - 20	8.5	273.09	28.30	0
20 - 30	9.0	258.20	26.09	0
30 - 40	9.5	244.81	24.18	0
40 - 50	10.0	233.58	22.62	0
50 - 60	10.5	222.63	21.15	0
60 - 70	11.0	212.67	19.84	0
70 - 80	11.5	203.56	18.68	0
80 - 90	12.0	195.43	17.66	0
90 - 100	12.5	188.28	16.79	0

Table 3.2: Change-driven CR scheduling policy.

To illustrate the effect of change-driven CR, Fig. 3.10 shows the system step response with different rise time: $t_r \in [8.5, 12.5]$ with a step of 1 second. First, if attackers send a single command to increase the speed from 10 to 100 mph ($\Delta x = 90$ mph), change-driven CR adopts PID parameters that result in $t_r = 12.5s$. The IDS will have enough time to detect and alerts will be issued at $4.0s$ when the vehicle speed is around 60 mph. Second, if attackers are aware of the CR module and want to avoid the additional response latency, they may choose to send 90 commands with 1 mph change in each message. This will generate a spike on the network traffic, which will also be easily identified by the IDS who monitors the CAN bus traffic. For commands with drastic changes in the setpoint, CR provides enough time for the IDS and system operators to respond or forces the attacker to take more effort (by sending more packets). Compared to the strategy where we increase t_r to $12.5s$ regardless of Δx, change-driven CR makes the system more responsive for

operations with smaller setpoint changes. Hence, change-driven CR greatly facilities the IDS to detect setpoint attacks and prevents drastic changes to take place within a short time.

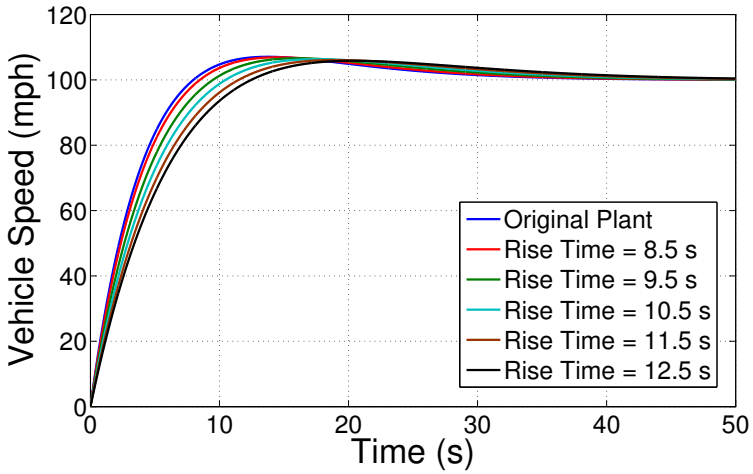


Figure 3.10: Step response of cruise control with change-driven CR.

3.5.4 System with Criticality-driven CR

$\lvert y(t) - y_c \rvert$ (mph)	t_r	K_p^{cr}	K_i^{cr}	K_d^{cr}
80 - 100	8.5	273.09	28.30	0
60 - 80	9.5	244.81	24.18	0
40 - 60	10.5	222.63	21.15	0
20 - 40	11.5	203.56	18.68	0
0 - 20	12.5	188.28	16.79	0

Table 3.3: Criticality-driven CR scheduling policy.

Sample criticality-driven CR scheduling policy is shown in Table 3.3. The vehicle becomes slower in response as its speed $y(t)$ approaches the critical point y_c. Both the

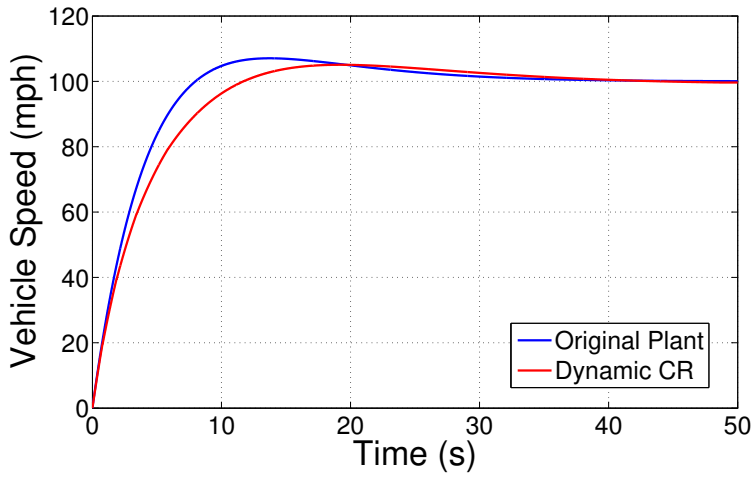


Figure 3.11: Step response of cruise control with criticality-driven CR.

QoS requirements and the IDS timing requirements are satisfied. The impact of criticality-driven CR on the cruise control step response is shown in Fig. 3.11 (the vehicle speed increases from 0 to 100 mph). Criticality-driven CR (with the policy in Table 3.3) increases t_r from $8.0s$ (original system) to $11.71s$. Consider an attack scenario where the vehicle is operating at 70 mph, the attacker sends a false actuation signal to wide open throttle. Without criticality-driven CR, the car will get to 100 mph in less than $8.5s$. However if criticality-driven CR is in effect, t_r will become $11.5s$ and IDS alerts will be issued at $3.0s$. If the attacker issues multiple malicious setpoint or actuation commands with smaller adjustments, the unusual traffic spike within a short time will also be obvious to the IDS. Both setpoint attacks and actuation attacks can lead to a change in $y(t)$, so criticality-driven CR can be used against both attack cases.

Change-driven CR and criticality-driven adjust the system from different perspectives: one focuses on the degree of change while the other focuses on the criticality of the current operating state. In practical cases, both strategies can be combined together to better harden the security protection of a CPS. For example, if the vehicle is operating at 85 mph ($|y(t)-y_c|$ = 15 mph) and the attacker issues a false command with the setpoint of 100 mph

(Δx = 15 mph), criticality-driven CR will dominate the CR module because it considers a more dangerous situation with t_r = $12.5s$ (versus t_r = $8.5s$ suggested by change-driven CR).

3.6 Discussion

CR prevents malicious attacks by slowing down on-demand the system response to sudden changes. When the desired change is not from an attack, this slowing down increases the latency for the system to reach the desired state. In such cases, the additional response latency is the cost of our approach. We note that this occurs rarely, in particular for criticality-driven CR, because the system response time during normal operation is left largely unchanged.

We currently adopt a greedy algorithm (the smallest norm-2 distance to the last selected PID parameter tuple) to pick a PID parameter tuple from a set of results that satisfy the design requirements of the original system. No new design requirements (such as QoS) need to be addressed. We could also choose to optimize one or more specific design requirements, such as minimizing rise time, settling time, and overshoot) and select the PID parameters accordingly.

As alternative to CR one may want to consider network-level IDS approaches. For example, DPI (deep packet inspection) could be used to identify setpoint attacks by inspecting the application-layer data payload. Overly aggressive setpoint-change commands would be simply rejected by the IDS, thus providing a seemingly simple protection scheme against such attacks. Unfortunately, intelligent attackers would tailor their commands and send sequences of small-change commands. The IDS would then have to rate-control these commands to protect the plant, which is fundamentally no different than CR. In addition, criticality-driven CR can prevent stealthy attacks that cannot be detected by the DPI, *e.g.* malicious setpoint commands with smaller setpoint changes sent over a longer period of

time. In addition to being at least as effective as DPI, CR provides an integrated analysis framework that allows to safely determine the parameters of the CR modules to protect the system while maintaining its operational requirements.

3.7 Related Work

Security issues of CPS can be tackled from two perspectives: communication network approaches and system-theoretic approaches [65]. Both are critical to secure CPS: communication network approaches can effectively prevent or thwart cyber-attacks from getting into a CPS while system-theoretic solutions rely on physical plant dynamics and system states to identify security-important sensor nodes and malicious behavior.

3.7.1 Communication Networks Approaches

Communication networks approaches enhance security protections from SCADA networks. Protection mechanisms have been proposed to secure critical infrastructures from intrusion detection systems (IDS) [18,28,30,41,42,66], path redundancy [44,45], authentication [43,46] and encryption [47,67], etc.

3.7.2 System-theoretic Approaches

System-theoretic approaches assume attackers have already compromised some physical components or malicious packets have already been inserted into the system. These solutions rely on the physical process dynamics to detect different kinds of cyber and physical attacks. However, the models are constructed based on approximations and subject to noise, which can result in a deviation of any model to the reality.

Several integrity attacks, including false data injection attacks (FDIA), reply attacks, stealthy attacks, and DoS attacks have been discussed in [68,69]. FDIA [51,54], which target against state-estimation and can circumvent the bad data detection mechanisms in the electrical grid, have been well studied in the context of smart grid. Liu et. al [51]

showed how FDIA can be staged; Xie et. al [52] showed that FDIA can lead to profitable financial misconduct in electrical market; and Kwon et. al [54] focus on three kinds of stealthy attacks and provide conditions under which attacks would succeed without being detected.

Replay attacks have also been discussed in [55]. The authors analyze attack models inspired by Stuxnet. Several defense mechanisms have been proposed, among which dynamic watermarking [48–50] can actively detect such attacks. It tries to identify malicious sensors by injecting small excitation into the system and detecting ones that report false sensing values. While a number of approaches [50, 51, 55] develop statistical hypothesis tests that detect attacks with a certain error rate, dynamic watermarking [48, 49] ensures that the amount of distortion that attackers can add without exposing their presence can have an average power of only zero.

3.8 Summary

Cyber-attack incidents against cyber-physical systems have shown that an adversary can make drastic changes to a plant within a short time. To deal with this problem, we introduce adaptive Commensurate Response (CR) as a cross-domain technique that protects a control system through appropriate control parameter selection against attacks coming from the communication domain. We provide detailed study on both change-driven CR and criticality-driven CR. Our case study on automobile cruise control demonstrates that change-driven CR can be used against setpoint attacks and criticality-driven CR is effective to combat both setpoint and actuation attacks. CR can effectively improve the system resilience and attack survivability and facilitate other defense mechanisms such as firewalls and IDS to better protect CPS.

4. TOWARDS IMPROVING DATA VALIDITY OF CYBER-PHYSICAL SYSTEMS THROUGH PATH REDUNDANCY*

4.1 Introduction

Cyber-Physical Systems (CPS) leverage communication technologies to monitor and control sensors and actuators in a physical system. The communication system used in CPS is called Supervisory Control and Data Acquisition System (SCADA). SCADA performs important functions in many national critical infrastructures such as power grid, oil/gas, water/sewage, and building automation. With the increasing need of remote and convenient control of distributed control devices and equipment, SCADA networks have started to incorporate Wide Area Network (WAN) and IP technologies to facilitate the Command and Control (C&C) process.

However, the incorporation comes at a cost: it creates new threats from potential cyber-attacks. Since SCADA controls many *safety-critical* sectors, its failure can cause irreparable damage to the physical system or jeopardize people who depend on it. Cyber security of CPS has attracted a tremendous amount of attention lately due to recent cyber-attacks that have been publicized in the news. To illustrate, in 2010, the Stuxnet worm sabotaged Iran's nuclear facilities through infecting Siemens S7 software and PLCs which control motor's spinning frequency [8, 70]. The attacker issued false commands from the PLC to periodically modify motor's spinning frequency while reporting normal sensing values back to the user. The recent Ukraine power outage [9] was the world's first power outage caused by hackers. The hackers compromised PLCs and leveraged them to issue unau-

*Part of this section is reprinted with permission from "Towards Improving Data Validity of Cyber-Physical Systems through Path Redundancy" by Z. Zheng and A. L. N. Reddy, 2017. *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security (CPSS)*, April 2017, Page 91-102, ©2017 ACM.

thorized commands to open circuit breakers. The power outage impacted approximately $225,000$ customers over three distribution areas [9, 71].

Attacks toward CPS normally come from two sources: through *physical access* or *remote intrusion*. Recently, an increasing number of cyber-attacks infect CPS through remote intrusion where attackers gain remote access by infecting corporate network as the first step. Since SCADA networks are usually connected with corporate LANs, the adversary may first employ stealthy Malware or viruses to infect workstations or web servers, stealing VPN confidentials and gaining access to the control network.

Next, attackers may compromise devices and equipment in the control network to launch attacks. SCADA system uses distributed *Programmable Logic Controllers (PLCs)* and *Remote Terminal Units (RTUs)* to control and monitor different types of *Physical Devices (PDs)* such as sensors, valves, pumps, drives, boilers and generators. Typically, each PD is only connected with one PLC/RTU. If a PLC/RTU is compromised, the attackers could directly send malicious commands to the connected PDs, causing unexpected changes to the physical system. They could also deceive the SCADA Server with invalid sensing values, compromising the data integrity of CPS networks. Such two attacks, called *false command attacks* and *data integrity attacks* respectively, could lead to widespread, costly and hazardous damage to an industrial control system, as illustrated by Stuxnet [8]. Compared with the SCADA Server, PLCs and RTUs are more susceptible to malicious attacks: they are normally embedded devices with limited processing and storage capabilities and weak or no security protections such as authentication, encryption, and antivirus capabilities. Therefore, PLCs and RTUs could become *Single-Point-of-Failures (SPOFs)* of SCADA networks.

Some recent work [45,72–74] has looked at redundancy to improve the control network resilience and to eliminate SPOFs. Peer-to-peer (P2P) overlay network was introduced to SCADA networks [45]. The paper [45] proposed a middleware-based approach to transmit

packets between the SCADA Server and PLCs in a P2P network. Their approach is able to protect networks from node crashes and data integrity attacks that are located between the source and destination. However, they cannot detect/prevent attacks from a compromised PLC. Other works [72–74] suggest to employ redundant hardware components such as servers, substations, controllers, and buses. This requires significant changes to the current system architecture and the additional hardware cost makes the solution difficult to deploy.

Intrusion Detection Systems (IDS) [32,75] and anomaly detectors [27,35,39] are effective in preventing and detecting malicious/malformed packets, but most work is focused on the Server-PLC channel. It is more difficult to monitor the PLC-PD traffic because: (1) most PLC-PD channels adopt serial links (*e.g.*, RS-485 and RS-232); and (2) the variety in PDs' vendors and applications affects traffic behavior. The lack-of-security in the PLC-PD channel makes PDs more vulnerable to cyber-attacks.

To improve the resilience and data validity of CPS, this section explores communication path diversity (especially on the PLC-PD channel) and proposes Path Redundancy to allow redundant sensing paths between the Server and a PD. The redundant paths go through other existing PLCs and are only used for sensing (data acquisition) purposes. The redundant sensing values are used to validate the data collected from the original path. When multiple values or inconsistent values are reported on the same *Physical Object* through redundant paths, this may indicate either (1) ongoing data integrity attacks from a single path or (2) false command attacks have been conducted on the Physical Device. We consider cyber-attacks coming from three locations: (1) a *single* compromised PLC; (2) Man-in-the-Middle (MITM) attacks originating between the Server and a *single* PLC; and (3) attackers inside the network but not MITM. Regardless of attacker's location, Path Redundancy can **prevent** data integrity attacks and **detect** false command attacks on a PD. Our non-intrusive solution takes advantage of existing PLCs in the CPS so that it

can be easily integrated into current deployments at minimal cost. Our solution is suitable for different types of CPS in terms of application environment and link types.

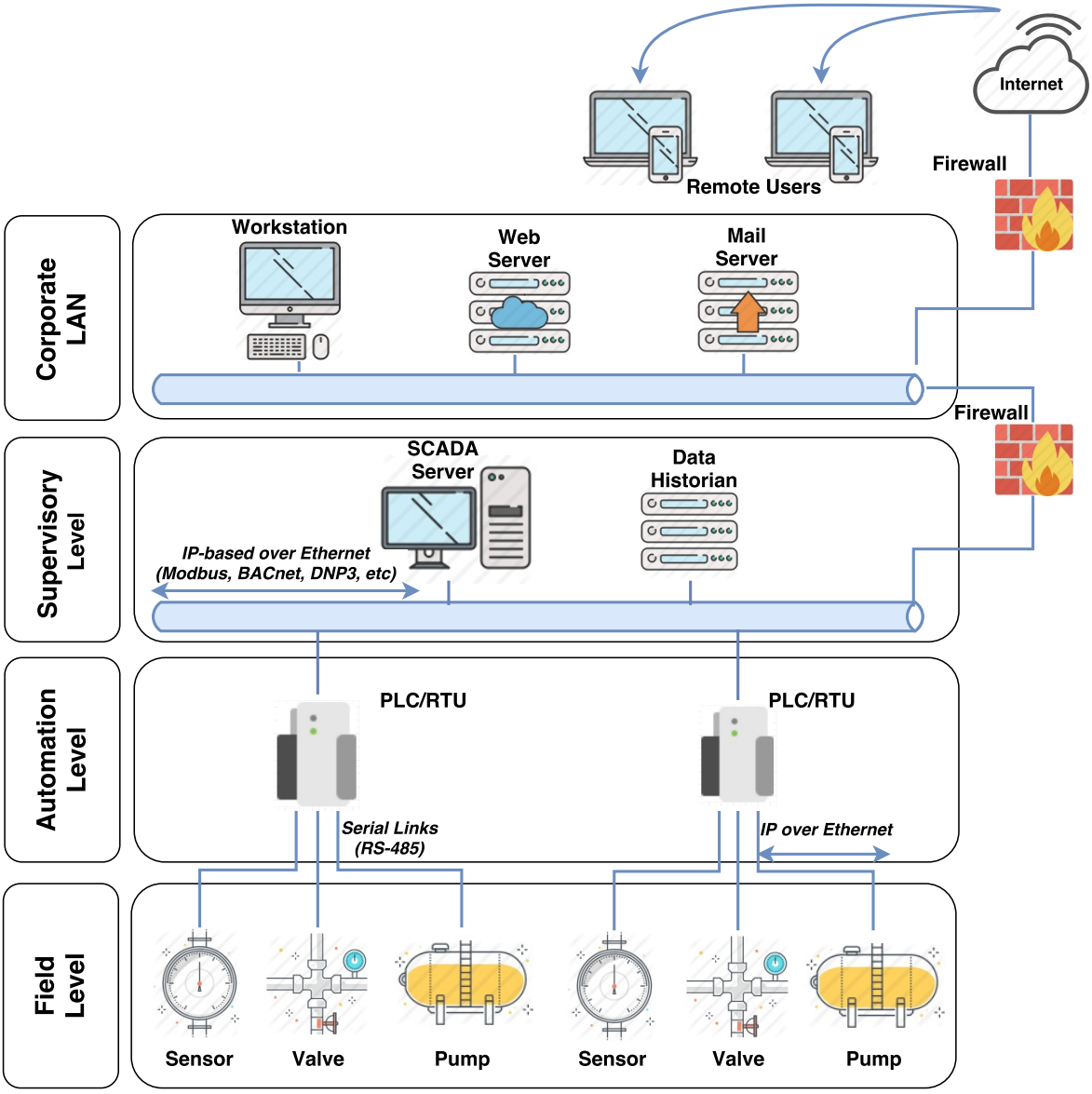


Figure 4.1: Architecture of a typical SCADA Network.

4.1.1 Contributions

The contributions of this section are the following:

- We propose Path Redundancy to enable redundant data acquisition paths between the Server and a Physical Device, which prevents a single compromised PLC from becoming the SPOF of CPS;
- We provide a detailed study on the implementation and demonstration of our solution in building automation networks;
- Based on our attacker model, our solution can effectively prevent ongoing data integrity attacks from a single path and detect false command attacks toward a Physical Device;
- Our solution is cost-efficient (in both computational and deployment cost), flexible, easy-to-integrate with existing deployments, and suitable for different types of CPS.

4.1.2 Section Structure

This section is structured as follows. In Section 4.2, we provide an overview of a typical SCADA system architecture and communication patterns. We also describe attacker models and the design requirements. We present the design of Path Redundancy in Section 4.3. Section 4.4 provides a detailed study on an application of our technique in building automation networks. The evaluation results are described in Section 4.5. Related work is presented in Section 4.6 and the conclusion as well as future work are provided in Section 4.7.

4.2 Preliminaries

4.2.1 SCADA Architecture

A generalized SCADA architecture is presented in Fig. 4.1. SCADA adopts a three-level hierarchical structure and usually connects with the corporate LAN. It contains four main components located at three levels: *Supervisory Level*, *Automation Level*, and *Field Level*, as described below.

Sensors and Actuators are directly interfaced to the plant or equipment. They are responsible for sensing physical conditions and make changes to these conditions. Sensors convert different physical parameters (*e.g.* temperature values, current and pressure) into electrical signals (analog or digital). Actuators are used to manipulate certain equipment such as water pump, valves and electrical relays. They are also called *Physical Devices (PDs)*.

PLCs and RTUs are geographically distributed in the CPS to conduct direct control and monitor of the physical plant. Typically, sensors and actuators are attached to a designated PLC or RTU, either wired (*e.g.*, through serial/Ethernet interface) or wireless (*e.g.*, through ZigBee [76]). PLCs and RTUs are basically "field control devices" that execute automation tasks based on inputs from sensors, control logic, or commands from the SCADA Server.

PLCs and RTUs perform similar functions. However, RTUs are more suitable for gathering telemetry data over large geographical areas because they normally use wireless communication; whereas PLCs are more suitable for local control, like assembly lines in factories because they are designed with multiple inputs and outputs. In this section, we use PLC to represent both field level controllers.

SCADA Server, or also called *HMI (Human-Machine Interface)* or *operator workstation*, interacts with PLCs and RTUs through IP encapsulated SCADA protocols to monitor

the physical process and issue control commands either autonomously or by operator interaction. The purpose of the Server is to acquire data from the field, store data points for historical purposes, present operators with graphical display of the state of the process, provide a channel for operators to manipulate the physical devices, and issue alarms when events happen.

Communication Channel is the medium for data transmission from one to another. SCADA contains two types of communication channels: the *Server-PLC channel* and the *PLC-PD channel*. The Server-PLC channel usually adopts IP encapsulated SCADA protocols based on various forms: wired (*e.g.*, through fiber optics or copper cable) or wireless (*e.g.*, through radio or satellite communication). Popular SCADA protocols include but not limited to DNP3 [4], Modbus [5], and BACnet [77]. The PLC-PD channel normally uses two types of media: (1) serial interface (*e.g.*, RS-485, RS-232, and RS-422) [78] with standard SCADA protocols or vendor's proprietary protocols; and (2) IP-based SCADA protocols over Ethernet.

4.2.2 Attacker Model

Attackers who have gained access to the control networks may launch attacks from different locations, depending on their penetration capabilities. Fig. 4.2 presents the attacker model considered in this section. We assume the SCADA server is equipped with adequate security protections and thus safe from cyber-attacks. We consider attacks from *three locations*: (1) a single compromised PLC; (2) MITM (Man-in-the-Middle) attacks originating between the Server and a PLC; and (3) attackers inside the network (not MITM).

4.2.2.1 A Single Compromised PLC

Attackers may infect a PLC using different approaches such as a USB flash drive and phishing emails containing Malware or viruses. Once the PLC gets infected, the attackers could take over the PLC by modifying its firmware or control logic, and use it to launch

attacks. In this section, we assume the adversary is only capable of compromising one PLC. Protection against multiple compromised PLCs is not considered in this section.

4.2.2.2 Man-in-the-Middle

This attack can be launched through several approaches: attackers may (1) infect a bump-in-the-wire device such as the network firewall or a switch between the two communication hosts, or (2) use ARP messages to trick hosts in the network to send their packets through the attacking host. Through either approach, the attackers are able to freely observe, intercept and manipulate unencrypted SCADA packets in transit. We assume the communication path between the Server and a *single* PLC can be compromised, while other PLCs are not affected.

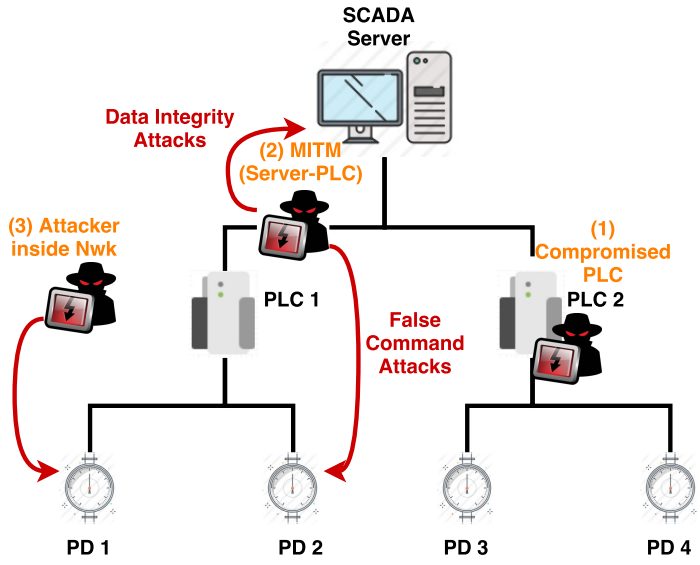


Figure 4.2: Attacker models considered in the section.

4.2.2.3 Attackers Inside the Network

This describes attackers who are able to launch stealthy attacks from somewhere in the network, but not MITM. In this case, attackers are able to send packets to either the Server or a PLC (by spoofing itself as a legitimate device), but are unable to observe/intercept packets in transit in the Server-PLC channel.

Regardless of the location of attackers, without sufficient security features, attackers are able to launch **false command attacks** and **data integrity attacks**. False command attacks refer to malicious control commands that are issued to a PD (usually an actuator or a drive) trying to cause dangerous consequences to the control system. Data integrity attacks describe false or untruly sensing values reported to the SCADA server on behalf of a PLC. Such attacks aim to deceive the Server with invalid system operational information. In particular, some attackers may combine replay attacks with normal sensing values to hide their ongoing false command attacks; while others may cause the Server to issue false control commands based on the false sensing data and the control loop. Therefore, our attacker model consists of six attack scenarios (two attacks at three locations).

4.2.3 Design Requirements

The goal of this section is to develop countermeasures against these types of attacks, improving the resilience and the data validity of cyber-physical systems. We are interested in a solution that can meet the following requirements:

- **Effectiveness.** The solution should be able to effectively thwart or detect different attacks with good accuracy. In should improve system resilience without introducing new threats or vulnerabilities;

- **Cost-efficiency.** Minimal computational and deployment cost is expected from both hardware and software aspects;

- **Practicality.** The solution should be able to be implemented and flexibly used in real-world CPS. Computational and storage capabilities of embedded devices, communication bandwidth, QoS (Quality-of-Service), and latency issues should be handled properly;

- **Easy-to-Deploy.** Since most SCADA networks have already been deployed or currently under deployment, our solution is expected to be easily integrated with current SCADA deployments with minimal change.

4.3 Path Redundancy Design

To improve the data validity of SCADA networks, we propose the Path Redundancy which enables redundant sensing paths between the Server and each PD. Fig. 4.3 illustrates the idea of our solution. For example, PD#2 has a unique communication path with the Server (Server–PLC#1–PD#2) used for both sensing and actuating purposes. In addition to the existing path, we enable an additional path as Server–PLC#2–PD#2 (the dashed orange lines in Fig. 4.3). This redundant path is *ONLY* used for sensing purposes, NOT for actuating. The redundant path goes through PLC#2 which is used to control and monitor PD#3 and PD#4. Similarly, PLC#1 is now responsible for sensing data points stored on PD#3 and PD#4 (the dashed red lines in Fig. 4.3).

Such change only affects data acquisition packets from the Server. All the other packets, including control and actuating packets, remain unchanged. Whenever the Server wants to acquire an object value on PD#2, it generates two sensing request packets to two different PLCs (one goes to PLC#1, and the other goes to PLC#2). The SCADA Objects used in both request packets are different, but they both reflect to the same Physical Object on PD#2. Both PLCs forward/reformat the request to PD#2 who responds with two independent packets to the Server through two paths. The Server receives two responses and uses the redundant data to validate the original. If the two values on the same Physical

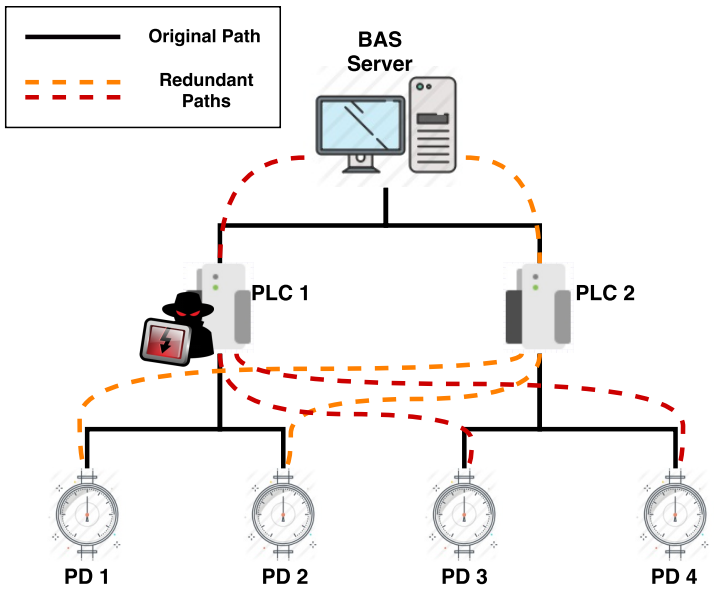


Figure 4.3: Redundant communication paths are enabled for sensing purpose.

Object are inconsistent or unexpected, it indicates that one PLC or communication path is compromised (since we assume only one PLC/path can be compromised). The Server can take further investigation to detect and isolate the compromised component.

The above example illustrates the basic idea of the **Path Redundancy**. While most of the description in this section describes only two paths, this idea can be generalized to more than two paths and quorum like policies or majority voting can be carried out to mask over incorrect values when more than two paths are available. Specific policies are determined based on different redundancy levels and constraints of the system. Typically, there are two main constraints: (1) limited processing and storage capabilities of PLCs, and (2) limited distance and node capacities of serial interface in the PLC-PD channel. Hence, we consider variety in two parts: *Number of Redundant Paths* and *Number of Data Objects*.

4.3.1 Redundant Paths

Based on specific application environment, vulnerability risks and redundancy levels, multiple redundant paths can be enabled to further strengthen the security protection and increase the confidence of data validity. Multiple redundant paths solution allows the system to automatically mask out the incorrect values and immediately identify the compromised path/PLC. This can greatly shrink incident response time and minimize potential damage impact on the physical system.

4.3.2 Redundant Data Objects

Since our solution suggests the use of existing PLCs to harden system security protection, adequate computing and storage capabilities are required at the Server and PLCs. Specifically, the support of multiple data replicas on the same Physical Object is required at the Server, and a PLC should be able to maintain data objects from the extended PDs. However, some PLCs or Server may not have enough storage or computational capability to support all the objects in the system. In this case, dynamic policies can be adopted which chooses a subset of Physical Objects for Path Redundancy. The subset of Physical Objects can be chosen based on their levels of criticalness and vulnerabilities. Many SCADA networks are capable of supporting the Path Redundancy with existing PLCs. Section 4.4 provides a detailed study of the situation in building automation networks.

For Path Redundancy with one redundant path, the PLC which is responsible for both sensing and actuating purposes toward a Physical Device is called the **Primary PLC** of that Physical Device, and the communication path passing through the Primary PLC is called the **Primary Path**. The redundant PLC that is only used for data acquisition purpose is called the **Secondary PLC** of a Physical Device, and the **Secondary Path** refers to the communication path passing through the Secondary PLC.

The reason to avoid control packets in Path Redundancy is three-fold. First, the control

function of a PLC towards extended PDs increases the potential degree of impact when a PLC is compromised. In this case, it could be even more difficult to control and restore the system. Second, the ACKs of control requests normally contain insufficient information for the source to validate data validity. Finally, when duplicate control commands arrive at PDs, the additional buffering and cross validation of commands may be needed close to the PDs, making deployment potentially more difficult.

4.4 Case Study: Building Automation Networks

This Section illustrates the design and implementation of Path Redundancy in building automation networks. First, we introduce building automation networks and the current data acquisition method. Next, we describe implementation challenges. Finally we describe our modifications in both hardware level and software level.

4.4.1 Background of BAS

Building Automation System (BAS) is a distributed control system responsible for the heating, ventilation, air conditioning (HVAC), lighting, security, fire and access control of a "smart building". BACnet™ is the standard data communication protocol for building automation and control networks. It defines a collection of **objects** and **services** to facilitate the communication. Details of these concepts have been described in Section 2.

In a BACnet/IP network, the BAS Server communicates with multiple **Field Panels (FPs)** (also called PLCs) that are scattered in the network. Multiple BACnet Field Panels that are located close to each other form a subnet which is called a *Building Level Network (BLN)*. All BLNs in a BAS form the BACnet internetworks. Each Field Panel connects to several Physical Devices which form a local *Field Level Network (FLN)*.

In today's BAS deployments, BACnet/IP is widely used as the communication protocol in the Server-FP channel. Since broadcast messages are not allowed to pass across IP routers, each BLN employs a special Field Panel called *BACnet/IP Broadcast Management*

Device (BBMD) to handle broadcast messages. The BBMD in a BLN repackages the message as a unicast and transmits it to its peers in other BLNs who broadcast the message on their local BLNs. BBMD can be either a physically distinct device or integrated into a BACnet FP.

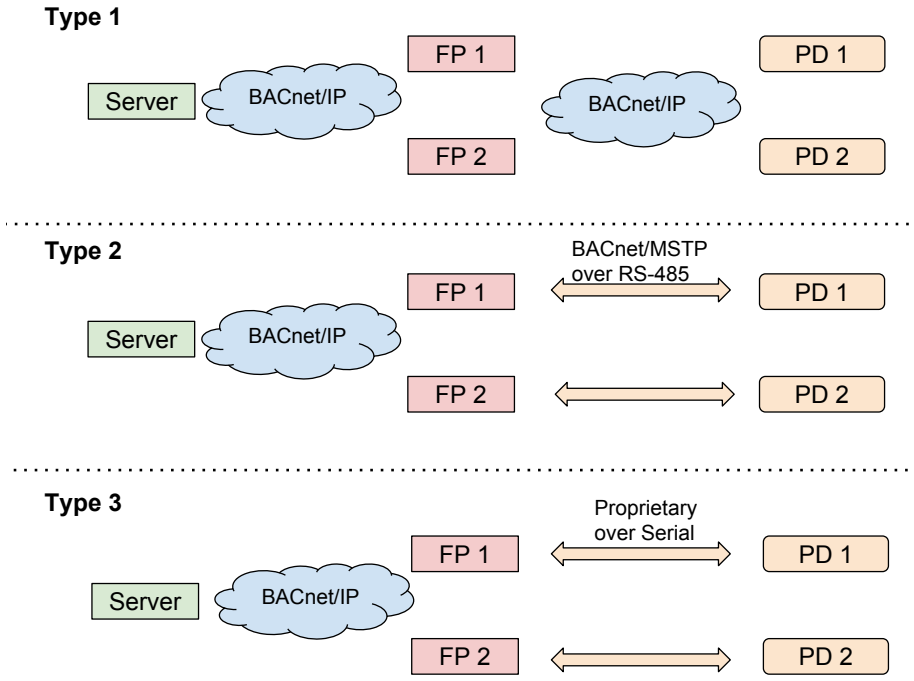


Figure 4.4: Three types of common FP-PD communication channels in current BACnet networks.

Three types of channels are found in the FP-PD communications, as shown in Fig. 4.4. These types are (1) BACnet/MSTP over serial links (*i.e.*, RS-485 / RS-232), (2) proprietary protocols or vendor-specific protocols (*e.g.* Siemens P1 Protocol) over serial links, and (3) BACnet/IP over Ethernet. The majority deployments adopt serial interface RS-485 in the FP-PD channel, and BACnet/MSTP is preferred over vendor's proprietary protocols because the use of proprietary protocols restricts the owner's flexibility and cost of replacement, service and switching to other vendors.

RS-485 [79] supports local networks and multidrop communication links. It can support up to 32 devices (nodes) spanning within $1,200$ m ($4,000$ ft). More devices can be connected using repeaters, up to the addressability limit (usually 256) of the devices used [80]. **Ethernet** over copper wiring supports up to 100 m (328 ft) whereas Ethernet over Fiber-optic can achieve a maximum distance of $80-100$ km ($50-62$ mi). The main drawback of RS-485 is that it only offers data transmission speed of at most 35 Mbps, whereas Ethernet can support the speed of $100-1,000$ Mbps.

When BACnet/MSTP or BACnet/IP is used in the FP-PD communication channel, the Field Panel acts like a BACnet Router and local controller; however, when proprietary protocols are used in the FP-PD channel, the Field Panel acts like a BACnet Gateway and local controller. In this case, there are two approaches to design these kinds of gateways:

(1). All of the physical devices' data points are aggregated into one "super BACnet device" that usually has a very large collection of BACnet objects.

(2). The Field Panel acts like a BACnet router to a "virtual network" of BACnet devices, and emulates the BACnet objects in each of those devices. In this case, there would be multiple distinct BACnet Devices in the gateway.

4.4.2 Current Data Acquisition Approaches

Data acquisition in a BAS usually takes two forms: (1) *Server-initiated* requests to retrieve data objects stored on a PLC; and (2) *FP-initiated* unsolicited response. Server-initiated method is usually used more frequently than the other. In the first method, the BAS Server usually sends ReadProperty or ReadPropertyMultiple services to a Field Panel according to different timers or commands from the system operator. The FP-initiated method, however, is usually used by a Field Panel to spontaneously report a Change-of-Value (COV) event. When a subscribed object value changes beyond the specified threshold, the Field Panel will send a ConfirmedCOV (CCOV) request to the Server to

report the COV event. CCOV messages are limited to the Server-subscribed objects and are considered as a substitute for Server-initiated data acquisition messages.

4.4.2.1 Hardware Level

If Ethernet is used as the communication interface, network switches are responsible for routing Ethernet packets. RS-485 communications normally require a hard wired, point-to-point cable connection between devices. It is designed for a master/slave topology where the master device polls each slave, wait for the response, and move to the next one. This architecture avoids data collision at the hardware level.

4.4.2.2 Software Level

If proprietary protocols are used in the FP-PD channel, the Field Panel caches object values of its connected Physical Devices in the local **Object Database**; whereas if the Field Panel acts as a BACnet router between BACnet/IP and BACnet/MSTP, it does not cache values, just forwards messages.

If a BACnet Field Panel maintains the local Object Database, it periodically updates the database by sending messages to its connected Physical Devices. When a Field Panel receives a BACnet read request, it is up to the vendor's application to determine how to prepare the response data. Two main types of mechanisms are found in BACnet Field Panels: **Response-with-Update (RWU)** and **Response-without-Update (RWOU)**.

Whenever a data acquisition request arrives at a Field Panel, the Field Panel of Type RWOU will directly respond to the request based on its local Object Database; whereas Field Panels of Type RWU will need to update the target content first before sending the response. Compared with RWU, RWOU minimizes the Round-Trip Time (RTT), but it may send stale data back to the Server, if the updating frequency of Object Database is low. It is up to the vendor's decision in choosing the design strategy, and the RWOU is relatively more widely used.

In addition, the vendor's software maintains an **Object Name Table (ONT)** that projects different Physical Objects to BACnet address pairs: <BACnet FP, BACnet Object ID>. The ONT is stored at all kinds of BACnet devices, including the BACnet Server, Field Panels, BBMD devices, and BACnet Physical Devices. Table 4.1 illustrates a sample ONT stored at the BAS Server. For example, "FL1_RM2_TEMP_SETPOINT" is mapped to $<$ FP$\#1$, AnalogValue$\#1$ $>$. Since each Physical Device is only controlled and managed by one Field Panel, this is a one-to-one mapping. Similarly, the ONT at a BACnet Field Panel/BBMD maps BACnet Objects to Physical Objects on its connected Physical Devices.

Physical Object Name	**BACnet FP**	**BACnet Object**
FL1_RM2_TEMP_SETPOINT	FP 1	AnalogValue 1
FL2_CHI_WATER_PRESSURE	FP 2	AnalogValue 2

Table 4.1: A sample Object Name Table stored at the BAS Server.

Therefore, if the system operator wants to acquire the current value of "FL1_RM2_TEMP_SETPOINT", the application at the BAS Server checks the ONT, interprets the Physical Object to the BACnet Object address pair (FP$\#1$: AnalogValue$\#1$), constructs the ReadProperty request with the BACnet Object, and sends the packet to the target Field Panel. Configurations and maintenance of the ONT is not part of BACnet Standard. Some vendors manually configure the ONT on each BACnet device during the initial setup period.

4.4.3 Challenges

In order to apply Path Redundancy to a BAS, we need to solve the following three challenges.

4.4.3.1 Hardware Level

The FP-PD communication channel can take two types of media (serial interface v.s. Ethernet) and two protocols (BACnet v.s. proprietary protocols). We need to develop hardware support for Path Redundancy in all cases, particularly for channels using serial interface.

4.4.3.2 Software Level

The BACnet devices employ Object Name Table for object naming translation. Current naming translation is a one-to-one mapping between Physical Objects and BACnet Objects. Path Redundancy requires a one-to-multiple ONT mapping.

4.4.3.3 Other Limitations

There are some other practical limitations that we need to take into account. For example, the capacity and distance constraints of RS-485 may limit the use of existing FPs for Path Redundancy.

4.4.4 Hardware Level Modifications

The hardware level modifications consider both Ethernet and serial links used in the FP-PD communication channel. Fig. 4.5 shows the changes for all three types of FP-PD channels. If BACnet/IP over Ethernet is used, no hardware modifications are needed. The Ethernet switch is responsible for routing packets to the target Physical Device. If BACnet/MSTP or proprietary protocol is used over serial links (RS-485 in this case), we need serial hubs/splitters. For example, if one redundant path is applied in a network: two FPs sensing values on a PD (cf. FP#1 and FP#2 in Fig. 4.5), we can connect a one-to-two serial hub to each FP and a two-to-one serial hub in front of each PD. In this case, packets from any FP can be directed to both PDs.

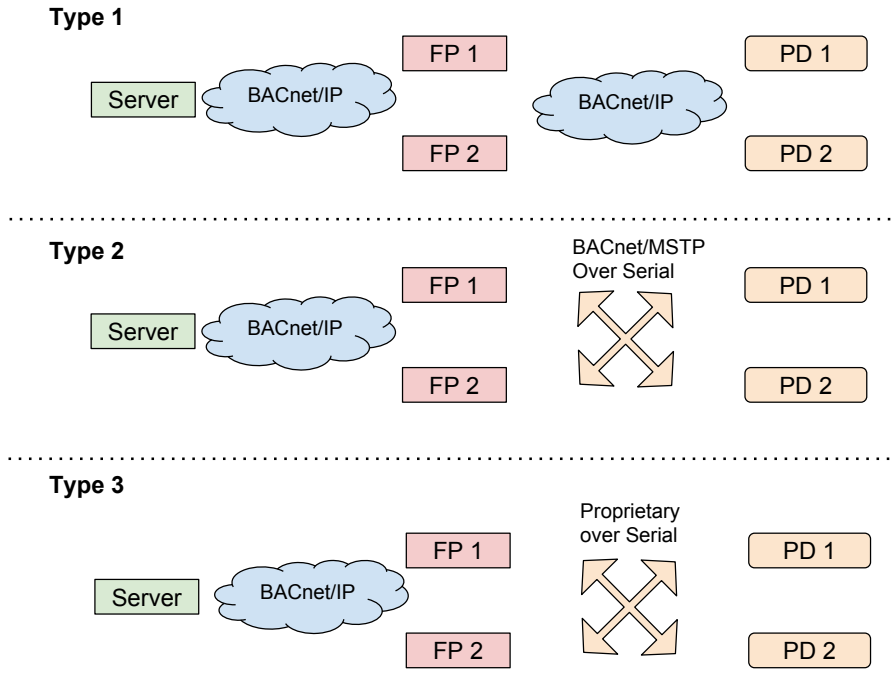


Figure 4.5: Hardware level changes for Path Redundancy.

4.4.5 Software Level Modifications

In the software part, we modify the Object Name Table stored at the Server and all BACnet Devices. Table 4.2 demonstrates the changes for the ONT at the BAS server when one redundant path is applied. For each Physical Object, the ONT includes two BACnet address pairs: <BACnet FP, BACnet Object ID>, one corresponds to the primary path, and the other corresponds to the secondary path. For instance, the "FL1_RM1_TEMP_SETPOINT" maps to <FP#1, AnalogValue#1> (the primary path) as well as <FP#2, AnalogValue#11> (the secondary path). The specific BACnet Object IDs on the redundant Field Panel can be any IDs that are not in use. When the Server wants to send a ReadProperty or a ReadPropertyMultiple request, the application interprets the Physical Object into two BACnet address pairs, constructs two packets, and further sends

out to two FPs. Similarly, the ONT at a FP/BBMD is also changed in the same way. Different vendors can use their own methods to maintain and update the ONT.

Physical Object Name	BACnet FP	BACnet Object
FL1_RM1_TEMP_SETPOINT	FP 1 (Primary)	AnalogValue 1
	FP 2 (Secondary)	AnalogValue 11
FL2_CHI_WATER_PRESSURE	FP 2 (Primary)	AnalogValue 2
	FP 1 (Secondary)	AnalogValue 22

Table 4.2: Modified ONT for Path Redundancy.

4.4.6 Practical Limitations

In a typical BAS network, each building (or a BLN) contains multiple FPs and a BBMD device. These FPs are located close to each other and their controlled Physical Devices. When necessary, RS-485 repeaters can be used to extend the distance and node (device) capacity.

Overall, our solution is applicable to all types of BAS FP-PD channels and it can be flexibly adapted to CPS with different kinds of constraints.

4.5 Evaluation

To evaluate the effectiveness and accuracy of our solution, we implemented Path Redundancy in an emulated Building Automation System using Raspberry Pi devices and a PC-based control server. We are interested to see if our solution can effectively protect the system from six attack scenarios described in Section 4.2.2 (two attacks launched from three locations).

4.5.1 Emulation Settings

Our testbed implements the Path Redundancy with one redundant path on all the objects. The testbed contains five components: the BACnet Server, a Primary FP, a Secondary FP, a BACnet PD, and an attacker. The attacker is a stand-alone device under the *MITM* and *Attacker inside the Network* case, but is mounted on a FP under the *Single Compromised FP* case. Fig. 4.6 shows the topology of the BAS testbed used in this work. The Server is emulated by a Ubuntu 16.04 system running on Intel Core i5-2320 3.00GHz CPU with 8G RAM. FPs are emulated by two Raspberry Pi 3 Model B devices, the PD is emulated by a Raspberry Pi Model B+, and the attacker is emulated by a Raspberry Pi 2 Model B. We assume BACnet/IP is used in both Server-FP and FP-PD communications. All the devices are connected using Ethernet within the same subnet, and the bandwidth is 1 Gbps.

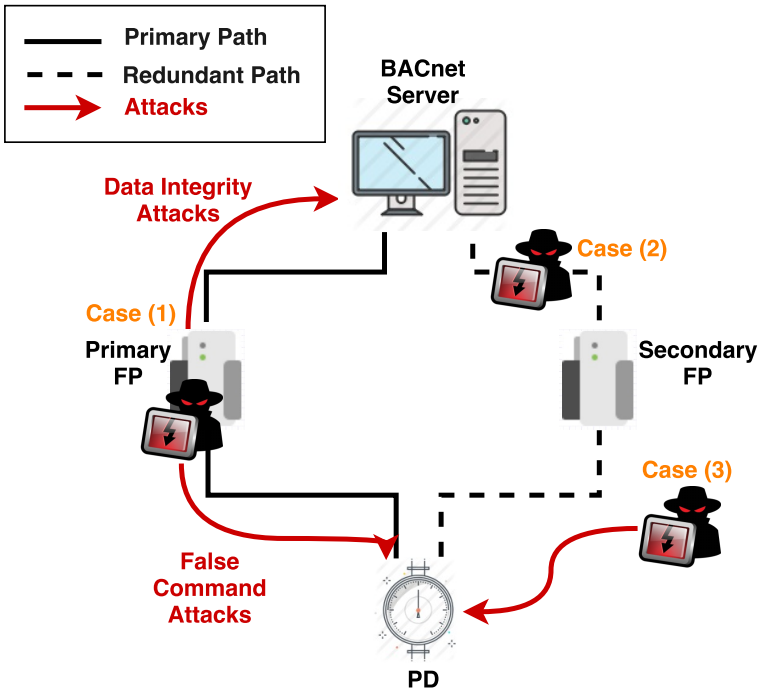


Figure 4.6: The topology of BAS testbed used in the work.

We modified the OpenSource BACnet Stack [81] to enable the FPs (here as BACnet Routers) forwarding packets between the BACnet Server and the PD. For BACnet/IP networks, no hardware modifications were needed, and about 800 lines of code were modified/added for the software modifications. Thus, our solution is easy-to-deploy with small modifications. After the changes, ReadProperty and ReadPropertyMultiple requests from the Server to the PD are routed through both FPs, while all other requests including WriteProperty and WritePropertyMultiple are only handled by the Primary FP.

First, without cyber-attacks, our emulation shows that the Server is able to receive consistent object values from two paths. Next, we emulated all six attack scenarios where the adversary launches both *false data injection attacks* and *data integrity attacks* from three different locations. We designed similar attack procedures for different attack scenarios, as described below:

1. The Server requests the current value of the interested object obj through *ReadProperty* packets, and the both FPs response the value of *obj* as *x*;

2. The attacker sends a malicious *WriteProperty* request to the PD to change obj from value x to value y under three cases: (1) *a compromised FP*, (2) *MITM*, and (3) *attacker inside the network (not MITM)*;

3. The Server sends another *ReadProperty* request to obj from two paths. The attacker under Case (1) and (2) manipulates the data response with false sensing value x, trying to hide the malicious changes made on the obj; the attacker under Case (3) does not manipulate *ReadProperty* response;

4. The Server receives inconsistent values (under Case (1) and (2)) or unexpected values (under Case (3)), which indicates the network is under attack.

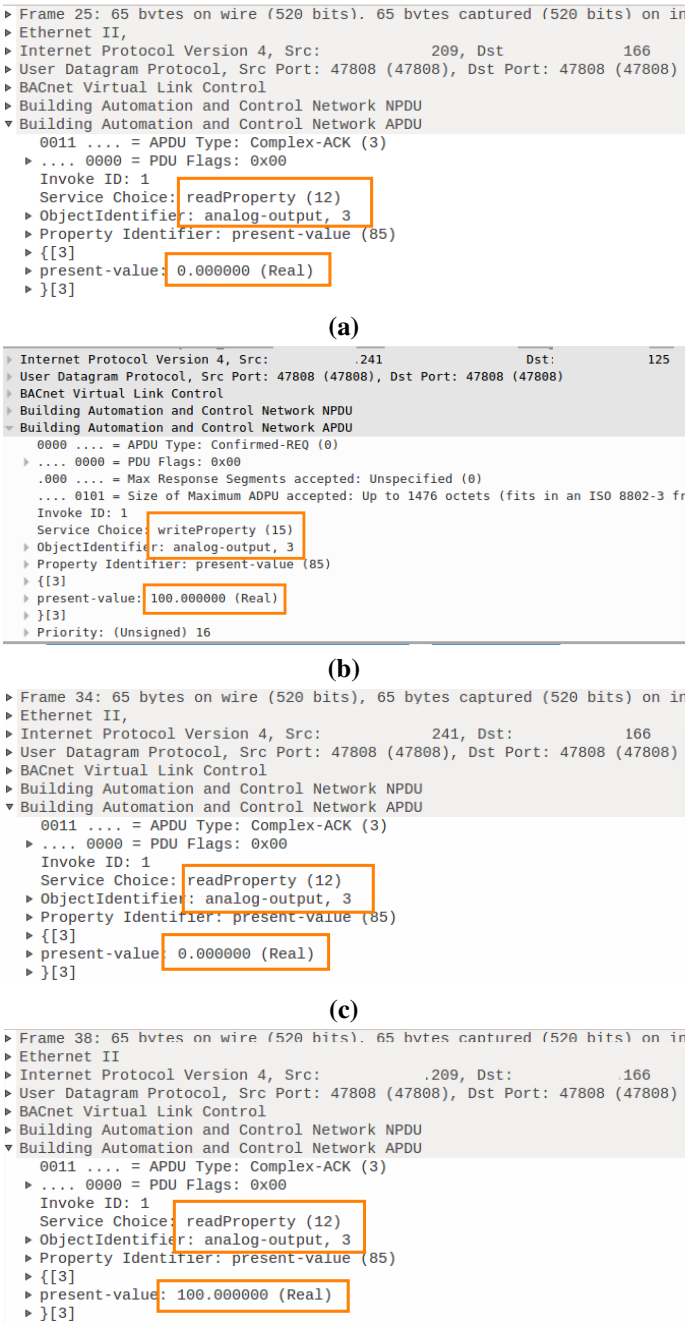


Figure 4.7: Emulation results of a compromised FP. (a) The Server reads AO#3 as 0.0; (b) The compromised primary FP changes AV#3 to 100.0; (c) The ReadProperty response from the compromised FP; and (d) the ReadProperty response from the good FP.

4.5.2 Detection Performance

Wireshark [25] was used to monitor the network traffic. Fig. 4.7 presents Wireshark snapshots of attack Case (1): *a single compromised FP*. Specifically, (1) The Server sends a *ReadProperty* request on object *AnalogOutput#3* whose current value is 0.0 (cf. Fig. 4.7-(a)); (2) The attacker compromises the primary FP and leverages it to send a *WriteProperty* request to the PD to change *AnalogOutput#3* to 100.0 (cf. Fig. 4.7-(b)); (3) The compromised FP reports false value of *AnalogOutput#3* as 0.0 (cf. Fig. 4.7-(c)), but the other FP responds with the real value as 100.0 (cf. Fig. 4.7-(d)); (4) the inconsistent values indicate one FP or path is compromised.

Table 4.3 presents that our solution can be used to prevent/detect different attack scenarios. No matter where the attacker launches the attack, when the adversary tries to deceive the Server with false sensor data from one path, our solution can **prevent** such attack because the redundant sensing value(s) can be used to validate the original value. When malicious false command attacks are launched toward Physical Devices, our solution can **detect** such attacks by either directly acquiring control objects/setpoints or acquiring the sensor values which are influenced by the changed control objects in the closed-loop physical dynamics. Our solution could effectively detect malicious and unauthorized attacks and help system operators to quickly take remedies.

Locations	Data Integrity Attacks	False Command Attacks
Compromised FP	Preventable	Detectable
MITM (Server-FP)	Preventable	Detectable
Attackers inside Network (Not MITM)	Preventable	Detectable

Table 4.3: Path Redundancy can effectively prevent/detect different attack scenarios.

4.5.3 Timing Performance

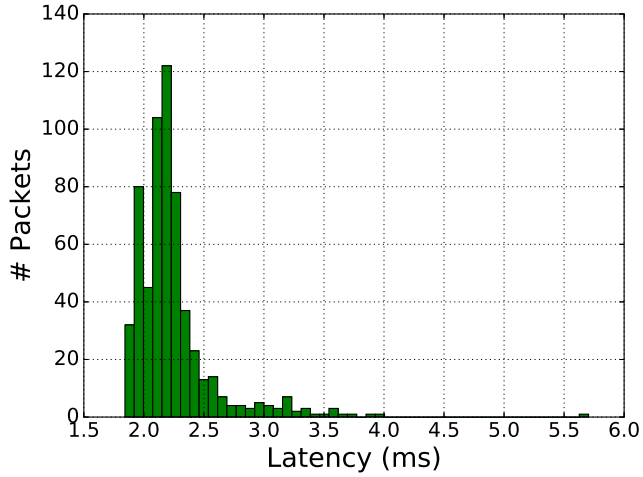
We conducted experiments to measure the latency of data acquisition on an object with Path Redundancy, and compared that with the original solution. We conducted 600 experiments in each case on our testbed. In Path Redundancy case, the Server sends two ReadProperty requests to both FPs who forward the packets to the PD and later forward the corresponding responses back to the Server. The latency of data acquisition consists of the RTT of all ReadProperty messages on the same object as well as the processing and scheduling latency from the devices.

Fig. 4.8 presents the distribution of data acquisition latency at the Server. The average latency increases from 2.24 ms in the original solution (cf. Fig. 4.8-(a)) to 3.71 ms in Path Redundancy with one redundant path (cf. Fig. 4.8-(b)). The additional 1.47 ms timing latency mainly comes from the processing and scheduling latency at the PD. Overall, the additional network latency is minimal (at miliseconds level) so that it will not affect the data acquisition process at the BAS Server.

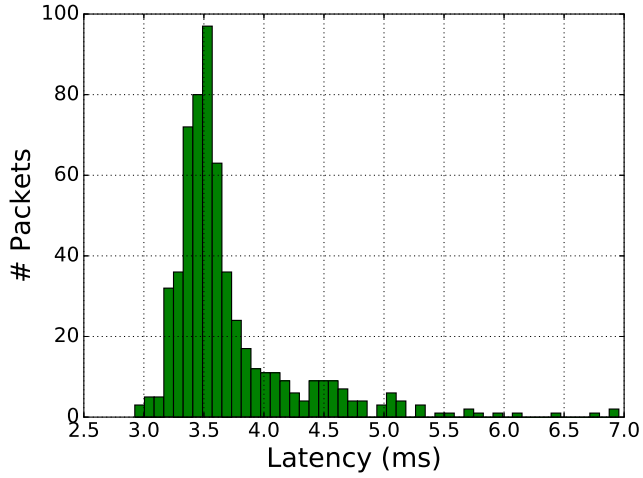
4.5.4 Network Traffic Overhead

Our solution only affects the Server-initiated data sensing packets. We measured the network traffic overhead in terms of **Packet Overhead** and **Bandwidth Overhead** based on our previous data collection from a building automation network in a university campus. In BACnet networks, the Server-initiated data sensing services are ReadProperty and ReadPropertyMultiple.

In terms of packet overhead, a single day traffic between the Server and all Field Panels contained $88,156$ packets. 23.56% packets corresponded to the read packets (20.22% ReadPropertyMultiple and 3.34% ReadProperty). When all objects are enabled for Path Redundancy, there will be 19.07% packet overhead for one redundant path and 32.03% packet overhead for two redundant paths.



(a)



(b)

Figure 4.8: Distribution of data acquisition latency from the BAS Server. (a) data acquisition without our solution; (b) data acquisition with Path Redundancy (one redundant path).

In terms of bandwidth overhead, a single day traffic contained 519 million bytes. 15.80% bandwidth (82 million bytes) corresponded to the read packets (11.56% ReadPropertyMultiple and 4.24% ReadProperty). When all objects are enabled for Path Redundancy, there will be 13.64% bandwidth overhead for one redundant path and 24.01% bandwidth overhead for two redundant paths.

It is noted that these volumes are low enough that increased bandwidth from Path Redundancy does not cause overload of the network.

4.6 Related Work

Several works have been conducted to improve the resilience and data integrity of SCADA system through redundancy. Some solutions suggest the use of **redundant hardware** (*e.g.* controllers, substations, PLCs or sensors) to provide redundant data replication [72–74]; while others focus on different types of **path redundancy** (P2P or pub/sub) to improve data validity [45, 82–85]. Since path redundancy does not require additional hardware devices, it is more cost-efficient and can be easily integrated with the current SCADA deployment practices. Different from our approach, previous path redundancy work focuses on the path diversity in the Server-PLC channels and thus is not able to detect a compromised PLC.

4.6.1 Redundant Hardware

The redundancies at hardware level and data replicas provide desired operation to CPS with failed nodes. Zhang et al. [74] present a general framework that abstracts the essential properties of sensor networks for the identification of compromised sensor nodes. They assume the network consists of a large number of redundant sensor nodes. Their application-independent approach does not need to consider dynamics of physical systems.

The authors in [73] describe the main deficiencies in the current communication networks of power grid and propose a new two-fold information architecture with various redundancy configurations. It adopts suitable computing and communication techniques to take into account the requirements of real-time data, security, availability, scalability, and appropriate Quality of Service (QoS). Multi-protocol label switching (MPLS), Vir-

tual Private Networks (VPN), and firewalls are applied to meet security requirement. This solution requires significant changes to the power grid in both hardware and software.

Cai et al. [72] develop a reliable remote control system for subsea blowout preventer stack. The remote control system is based on the off-the-shelf triple modular redundancy system and various redundancy techniques such as controller redundancy, bus redundancy and network redundancy. They show that when faults happen on PLCs, discrete input groups and analog input groups, alarms will be raised at the HMI.

4.6.2 Path Redundancy

The authors in [82] propose an agent-based layer on top of a P2P (peer-to-peer) network to improve message exchange reliability. It also discusses the advantages and disadvantages of using Chord network [86] for power grid applications.

Germanus et al. [45, 85] propose a P2P overlay network to increase the resilience of SCADA systems. The P2P communication between a MTU (Master Terminal Units) and multiple RTUs (Remote Terminal Units) provides path redundancy and data replication. A middleware is inserted between the SCADA application and the IP layer at every SCADA device. The middleware extracts SCADA payload and stores the data in the P2P overlay. When a MTU receives a message from SCADA network, it requests the same message from different replicas in the P2P network to validate the original message. Their approach can protect SCADA from node crashes and data integrity attacks that are located between the source and destination. However, they cannot detect malicious RTUs colluding on behalf of an adversary. Their intrusive middleware-based approach is scalable to accommodate ongoing developments of interconnected SCADA systems.

In contrast, our approach is shown to be able to detect a compromised PLC/RTU which may be used by an adversary to launch data integrity attacks and false command attacks. Compared with their intrusive middleware-based approach, our solution takes advantage

of the existing PLCs and equipment in the system and thus can be more easily applied to current deployment practices.

Germanus et al. [87] further proposed Coral, a decentralized P2P protocol that provides low latency and reliable convergecast for sensor data collection. Coral consists of three parts: (1) periodic link latency measurements, (2) latency minimal and disjoint path search, and (3) robust and reliable convergecast routing. Their protocol is evaluated using a case study: system protection workload on a realistic power transmission network topology.

The authors in [84] propose a publish-subscribe middleware framework to meet the data delivery and surveillance requirements for electrical power grid. Data streams produced at the source (publisher) are distributed to destinations (subscribers) without the publisher having to track all the subscribing entities. The pub/sub structure requires message broker nodes to manage forwarding and receiving mechanism. This framework is suitable for a sensing-only network where multiple devices are interested in the same data from the same device. It explores the path diversity between the publisher and its subscriber(s), rather than the path between the physical devices and the publisher. Thus, they are not able to detect/prevent attacks from a compromised subscriber/path. The section evaluated the performance in terms of forwarding latency and load scalability, but failed to provide evaluation on security performance or redundant paths.

4.7 Summary

Path diversity has been explored in many contexts before. This section shows that Path Redundancy is a simple and effective solution to counter compromised PLCs in CPS. The proposed redundant paths are enabled with existing PLCs and are only used for sensing purposes. We considered six attack scenarios and described a demonstration of our techniques in an emulated Building Automation System using Raspberry Pi devices and a PC-based control server. Emulation results have shown that our solution could effectively

prevent data integrity attacks and detect false command attacks from a single source. Our solution is cost-efficient, easy-to-deploy, and suitable for different types of CPS with different communication interfaces.

5.1 Introduction

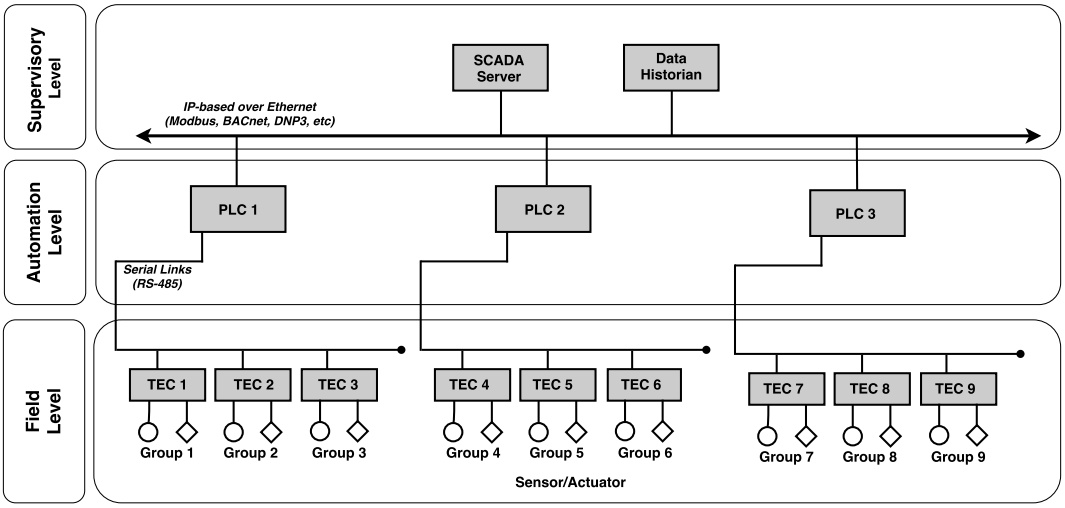


Figure 5.1: Traditional SCADA Architecture.

Cyber-Physical Systems adopt a hierarchical tree structure which can be divided into three levels: supervisory level, automation level and field level, see Fig. 5.1. The supervisory level consists of the SCADA Server and the data historian. The SCADA Server monitors the operating state of the system and applies control when necessary. The Server is connected to a number of embedded controllers called Programmable Local Controllers (PLCs) which make up of the automation level. Each PLC is connected with a number of field level devices such as Terminal Equipment Controllers (TECs) or Remote I/O (RIOs) using either a tree structure or a daisy chain structure (multidrop bus). PLCs are responsible for reporting sensing data to the SCADA Server, receiving control commands from the Server, and applying actuation to the TECs. At the field level, each TEC is directly wired

to one or more sensors and actuators. The TECs can be considered as the second-level controllers and they enable the communication capability of the sensors and actuators.

Communication takes place between two adjacent levels: the SCADA Server communicates with the PLCs who will be responsible to communicate with the TECs. Different SCADA protocols and communication interfaces can be used in the network. Ethernet is widely used as the interface between the SCADA Server and the PLCs, and the PLC-TEC communication channel can adopt either Ethernet or serial links (*e.g.* RS-485 and RS-232).

Due to the limited computing capability and memory capacity, many PLCs and TECs are normally not equipped with adequate endpoint security protections, which makes them susceptible to cyberattacks. To illustrate, PLCs were the attack target in the infamous Stuxnet [8] attack and the Ukraine power grid attack [9]. When a PLC is attacked by malicious attackers, the Server fails to sense and actuate the TECs controlled by it. Similarly, when a TEC is attacked, the sensors and actuators controlled by the compromised TEC will be affected. Thus, PLCs and TECs may become the Single-Point-of-Failures (SPOFs) of a CPS network.

In this section, we propose a new CPS architecture to overcome SPOFs. The new architecture turns the CPS from a tree structure to a general switched structure (fat-tree structure). It provides data replicas to check data validity and enables control switching when a controller instance is compromised. In addition, we introduce virtualization into the embedded controllers to minimize the number of hardware devices. The emulation using Raspberry Pi devices and Docker containers demonstrates that it is feasible to concurrently host 6 to 11 virtual controllers on a Raspberry Pi device. Given the real-world embedded controllers with better capability, this solution should be feasible to the real-world CPS controllers. Our solution can provide better security protection to the embedded controllers with minimal cost.

The rest of this section is arranged as follows. Section 5.2 discusses the solution of using redundant hardware devices to overcome SPOFs. Section 5.3 presents our new CPS architecture. Control switching strategy is discussed in Section 5.4 and evaluation is presented in Section 5.5. Finally, we summarize this section in Section 5.6.

5.2 Redundant Physical Controllers

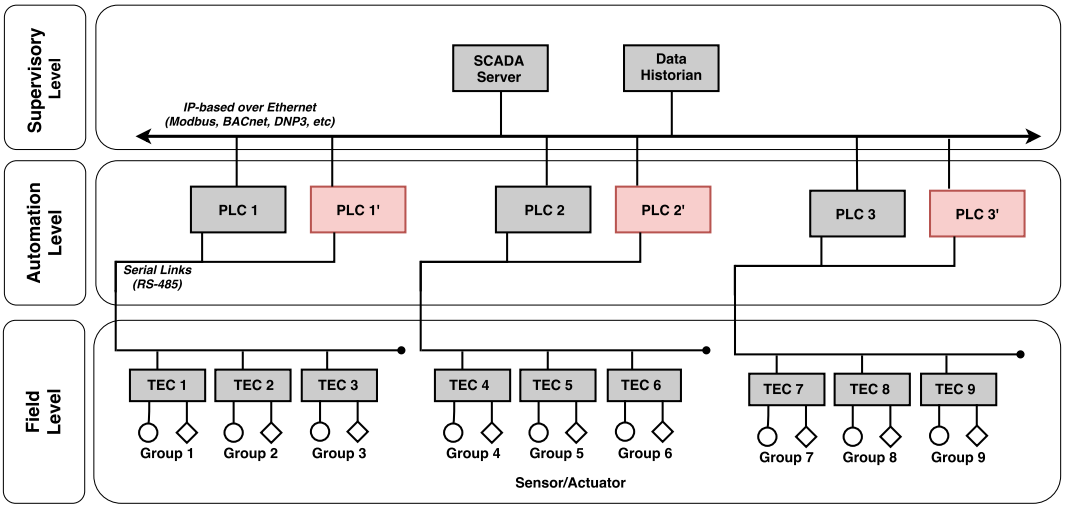


Figure 5.2: SCADA Architecture with Redundant Controllers.

To overcome SPOFs and improving system resilience, one possible solution is to employ redundant controllers to act as backups. As is shown in Fig. 5.2, each TEC is connected to an additional PLC (PLC 1', PLC 2', or PLC 3'). The original PLCs (PLC 1, PLC 2, and PLC 3) are responsible for both sensing and actuation. The redundant PLCs act as hot standby and are only used for sensing purpose. They provide additional replicas of the sensing values to facilitate the Server to validate data integrity. For example, when the Server wants to collect an object value on a TEC, the Server generates two independent read packets to two PLCs. Both PLCs collect the object value and send back to the Server independently. The Server then compares the two values and alarms will be raised if the

two values are inconsistent. Actuation commands are only handled by the original PLCs so that duplicate actuation from the same command can be avoided.

This solution is effective to overcome SPOFs and has been adopted in many large-scale and distributed networks such as electrical grids where data integrity is critical. However, this solution requires additional hardware devices and the level of redundancy is proportional to the number of redundant controllers: the more redundant controllers are deployed in a network, the higher level of redundancy it can provide. The additional hardware, space and management cost makes this solution not suitable for many deployment environments. As an illustration, the price of a PLC device ranges from several hundred Dollars to thousands of Dollars. For networks with a large number of controllers and dangerous security threats, the cost of redundant hardware controllers can become a huge burden for the network mangers.

5.3 New CPS Architecture

5.3.1 New Architecture Overview

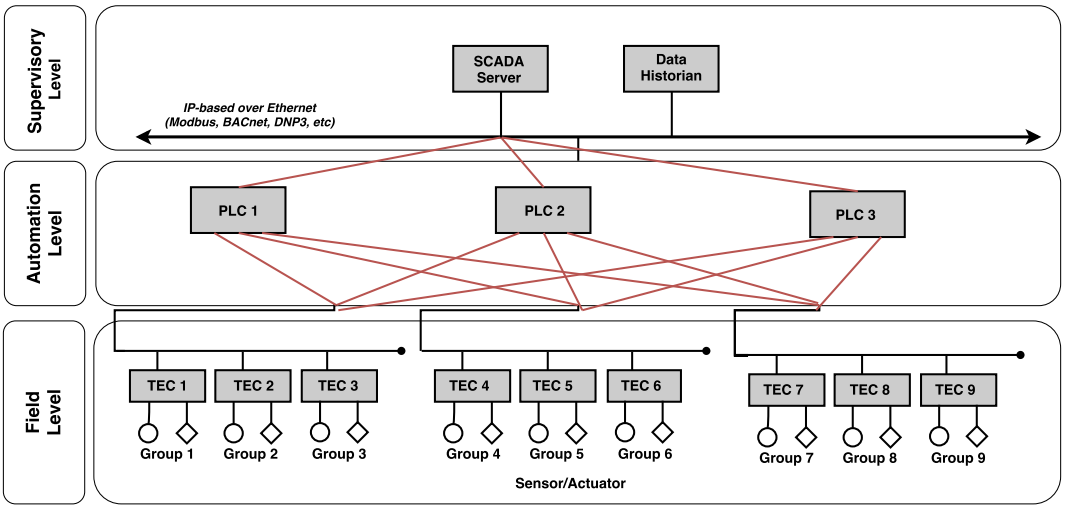


Figure 5.3: SCADA Network in Fat-Tree Structure.

We introduce a new architecture to the CPS network, see Fig. 5.3. It follows a fat tree structure where each PLC is connected to all TECs. For objects on a particular TEC, all PLCs are responsible for data acquisition, but only one PLC is responsible for actuation. Redundant sensing data can be used to detect compromised PLCs. When a PLC is identified as anomalous, the Server will isolate the compromised PLC from the network and designate another PLC to take over the control from the compromised device. This architecture provides data redundancy and enables control switching when a PLC is identified as anomalous. Compared to the redundant controller solution, this architecture minimizes the number of additional hardware devices needed in a network, making this solution more feasible to different deployment environments.

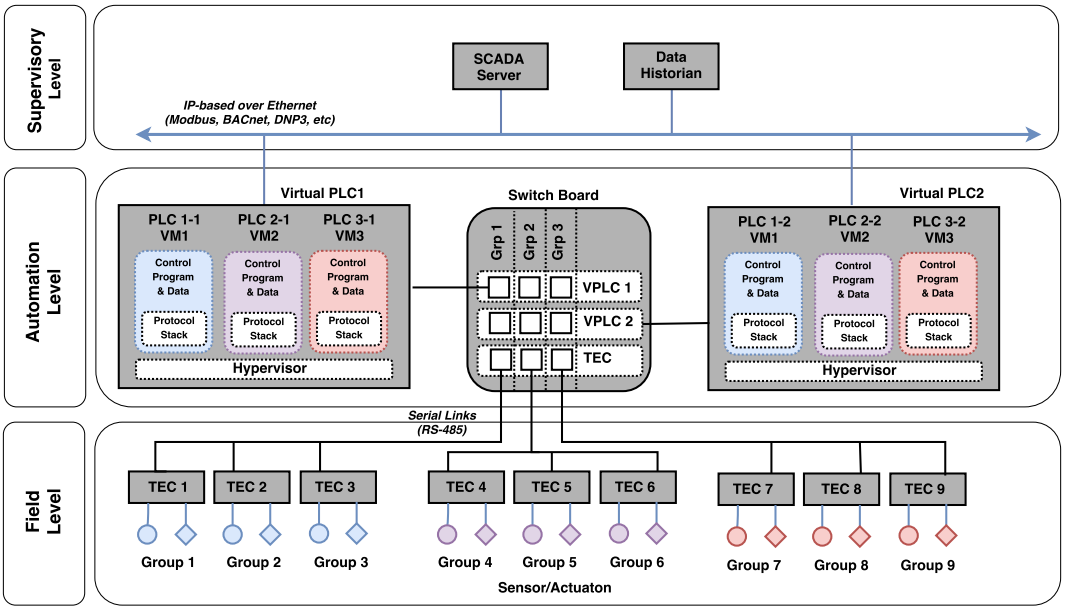


Figure 5.4: Fat-tree Structure at the Automation Level.

In order to concurrently run multiple independent control programs within a single hardware device, we introduce virtualization into the embedded controllers. The new

PLC with multiple virtual environments are called **Virtual PLCs (VPLCs)**. Each VPLC hosts multiple independent and isolated virtual environments (*e.g.* virtual machines), and each virtual environment contains the entire control context in a traditional PLC. This context includes but not limited to control programs, communication protocol stack, and data points. The hypervisor manages the scheduling and execution of the guest virtual machines.

Fig. 5.4 shows the network architecture of our proposed solution where we employ the fat-tree structure and virtualization at the automation level. Each VPLC contains three virtual machines, each of which contains the control program of some sensors and actuators. For example, PLC 2-1 and PLC 2-2, located on two different VPLCs, simultaneously monitor the sensors and actuators in Group 4-6.

Virtualization Degree (VD) represents the maximum number of virtual machines that can be hosted concurrently on a physical controller; and **Replication Degree (RD)** represents the total number of instances (replicas) of a controller in a system. A controller instance is represented by a pair of IDs $< i - j >$ where i denotes the controller ID and j denotes the instance ID of the Controller i. For example, VPLC 1 hosts three PLC instances (PLC 1-1, 2-1, and 3-1) that correspond to three different PLCs, and VPLC 2 contains the second copy of the three PLC instances (PLC 1-2, 2-2, and 3-2). The example in Fig. 5.4 demonstrates that we are able to provide two data replicas for each PLC with only two VPLCs (the VD is three). Compared to the original architecture (in Fig. 5.1) which requires three independent PLC devices, the proposed solution is able to provide redundant data replicas with fewer hardware devices.

The new structure and virtual controllers can also be incorporated at the field level controllers: TECs and RIOs. Fig. 5.5 shows the CPS architecture. The automation level adopts the same VPLCs as the ones shown in Fig. 5.4, where we adopt two VPLCs with the VD of three. At the field level, the traditional TECs are replaced by the **Virtual TECs**

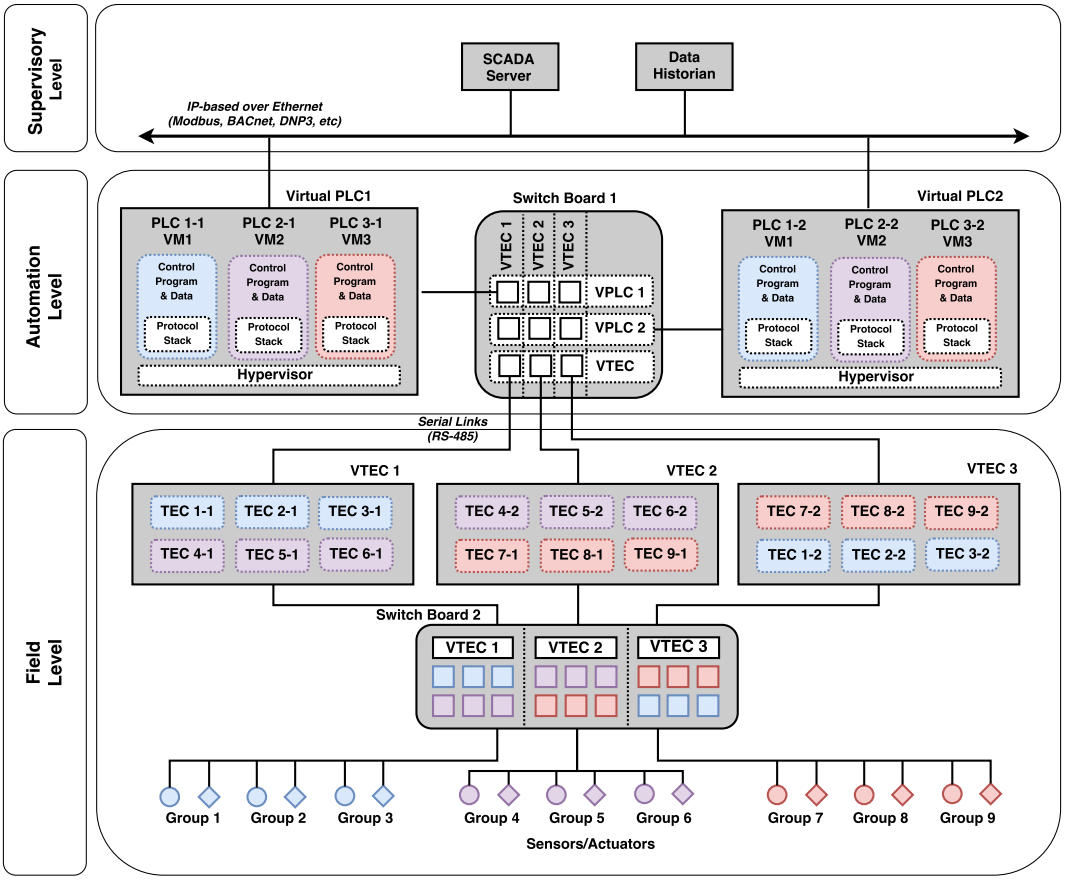


Figure 5.5: Fat-tree Structure at the Automation and Field Level.

(VTECs) with the VD of six. Therefore, instead of requiring nine TEC devices to control the sensors and actuators (as is shown in Fig. 5.4), the solution can provide two replicas for each TEC instance using only three VTECs. The VD varies among different types of controllers with different processing capabilities and memory constraints, and the RD can be determined based on the threat rick and the security threats of a network.

5.3.2 New Control & Communication Patterns

Sensing and actuation in a CPS network correspond to the read and write packets in a communication network respectively. The new architecture changes the control and communication method in both the Server-PLC channel and the PLC-TEC channel.

When the Server wants to query an object value, it sends multiple independent read packets to all the controller instances which have a copy of the targeted object. Each controller instance responds to the Server with the data stored in its local database. Once the Server receives all the data values, it compares all the data values of the same object and identifies the inconsistent data. The false data implies the compromise of the corresponding controller instance and the operator isolates the infected controller from the network. We assume only one controller instance may be compromised at a time. If all the data values of the same object are consistent, the Server considers the sensing data as valid.

Different from the read packets, our solution only allow one controller instance to handle the write packets. This is because duplicate write packets may result in duplicate actuation and the synchronization between different controllers need to be handled properly. When the Server wants to apply an actuation signal to an object, the write packets are only sent to one controller instance. Among all the controller instances of a sensor or actuator, the controller instance that is capable of both sensing and actuation is termed as the **Primary Controller Instance (PCI)** while all the other instances are called the **Auxiliary Controller Instances (ACIs)**. The ACIs act as the hot standby and are only used for sensing purpose.

All controller instances of a device are assigned to different physical controllers. In addition, the PCI for different devices are normally assigned to different physical controllers. For instance, as is shown in Fig. 5.5, the PCI for Group 4 can be TEC 4-1 (in VTEC 1) whereas the PCI for Group 5 can be TEC 5-2 (in VTEC 2). The allocation of the

PCI is based on CPU utilization, memory constraints, network bandwidth and port limitations. Even though a large number of Malware and viruses target at individual virtual machines (controller instances), some may also compromise the hypervisor which controls and manages all the virtual machines on a device. The distributed allocation strategy makes the attacks that try to compromise all controller instances more difficult to succeed.

5.3.3 Virtualization Degree

The Virtualization Degree represents the number of virtual machines that can be concurrently hosted on a physical controller. Given the RD and total number of control programs in a CPS network, the number of virtual controllers that are needed in a network is depended on the value of VD:

$$N.O.\ of\ Virtual\ Controllers = \frac{RD * N.O.\ of\ Control\ Programs}{VD} \tag{5.1}$$

Thus, we are interested to estimate the VD that can be supported in a CPS controller.

Many CPS networks employ automated process to periodically poll data and apply actuation. Data acquisition and actuation take place between different types of devices. It can happen (1) between the SCADA Server and the PLCs; and (2) between the PLCs and the TECs. Fig. 5.6 shows a sample timeline of the PLC-TEC communication channel where the PLC periodically applies sensing and actuation to the TEC through ReadProperty and WriteProperty packets. Each period contains the following steps. First, the PLC initiates a ReadProperty Request to the TEC to query an object value; second, the TEC receives the request, fetch the target object value from its database, and sends the value back to the PLC; third, the PLC estimates the current operating state based on the sensing value; next, the PLC sends a WriteProperty Request containing the actuation signal to the TEC; finally, the TEC receives the WriteProperty Request, applies the control, and

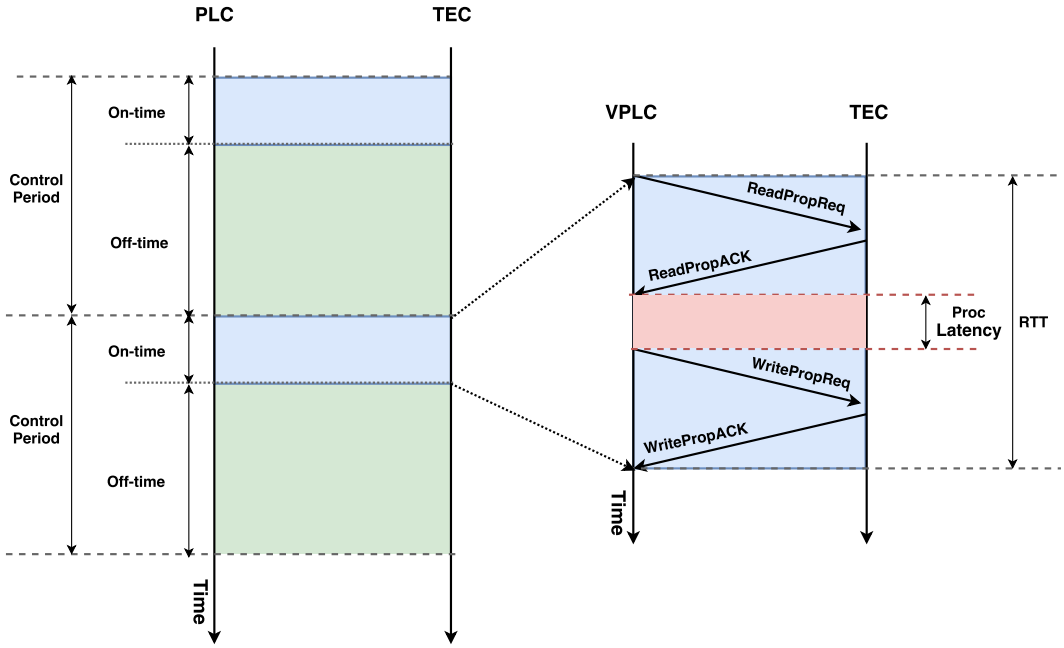


Figure 5.6: The Timeline of the PLC-TEC Communication Channel.

replies the request with an ACK packet. Since the Control Period is normally larger than the total transmission and actuation latency, we model such automated control process as an on-off model. As is shown in Fig. 5.6, the Control Period represents how frequent the control cycle is applied, and it can be separated into the on-time and the off-time. Packets are transmitted during the on-time and the PLC switches into the sleep mode during the off-time. The on-time can be further divided into the propagation delay and the processing latency. In this model, each Control Period contains the propagation delay of four packets, the processing latency at the PLC and at the TEC device.

The virtual controllers (VPLCs and VTECs) utilize the off-time of one controller instance to execute the control program of other controller instances, so that they can execute multiple control programs within a Control Period.

The value of VD can be estimated by two methods. First, we assume the timeline of the VPLC is divided into continuous Control Periods, and the VPLC executes the con-

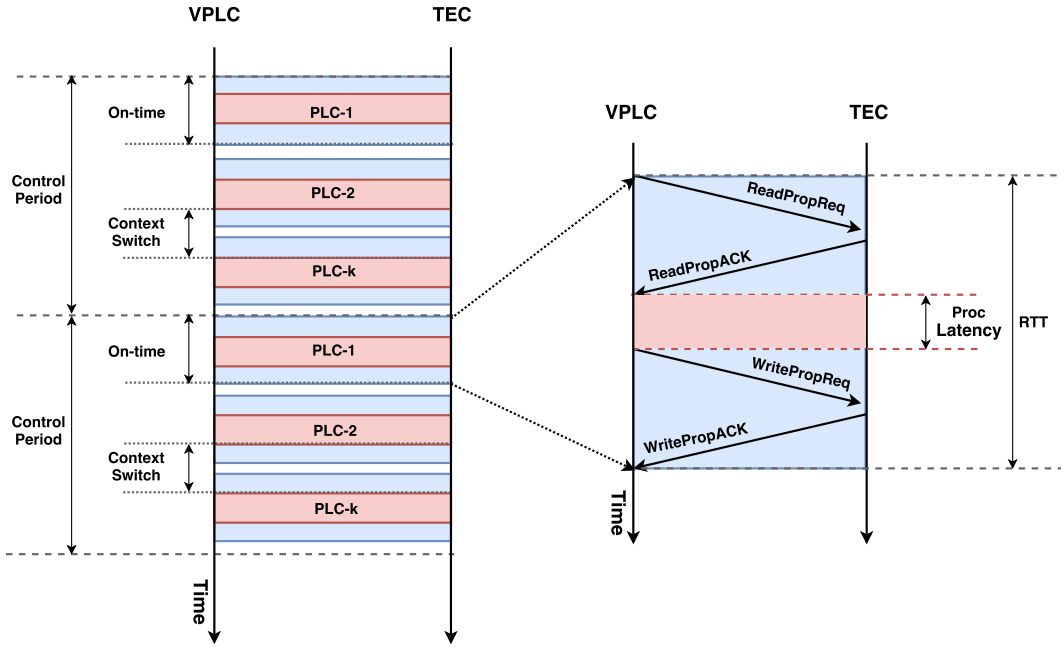


Figure 5.7: The Timeline of the VPLC-TEC Communication Channel.

troller instances one after another within each Control Period. Fig. 5.7 shows the sample timetime between the VPLC and the TEC. Essentially the virtual controller divides each Control Period into several on-times where each on-time is allocated to execute the control program of one controller instance. As is shown in the figure, the VPLC is able to run k PLC instances in a Control Period, while making sure that each controller instance can be executed periodically. We assume that all controller instances have the same on-time and off-time. Let k represents the VD, t_{cp} represents the Control Period, t_{on} represents the on-time, t_{off} represents the off-time, t_{proc} represents the processing latency at the VPLC, k_{cs} denotes the context switch latency, and t_{prop} represents the total propagation delay for the ReadProperty and the WriteProperty packets in a Control Period.

The controller instances within a physical device are executed one after another within

a Control Period. In this case, the VD k is bounded by:

$$k \leq \frac{t_{cp}}{t_{on}} \tag{5.2}$$

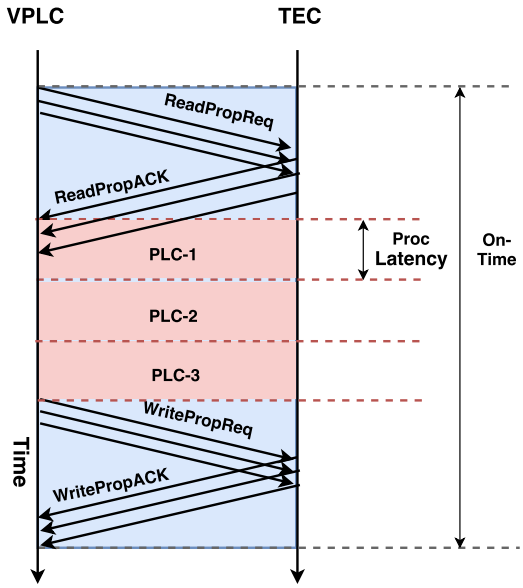


Figure 5.8: The messages within a Control Period are pipelined to transmit.

In the second estimation method, the virtual controller pipelines the packets that need to be sent from different controller instances and sends the messages together, see Fig. 5.8. This method can greatly minimizes the overall transmission time within a Control Period. If each Control Period can run k control programs, the total Control Period t_{cp} contains the propagation delay of a ReadProperty and a WriteProperty packet ($k * t_{prop}$), the processing latency of k control programs ($k * t_{proc}$), and $3 * (k - 1)$ times context switch latency ($3 * (k - 1) * t_{cs}$). In this case, the value of VD k can be estimated as:

$$k * (t_{prop} + t_{proc}) + (3 * (k-1)) * t_{cs} + \leq t_{cp} \tag{5.3}$$

$$k \leq \frac{t_{cp} + 3 * t_{cs}}{t_{prop} + t_{proc} + 3 * t_{cs}}$$

The first method is applicable for both the Ethernet and the serial links. The second estimation method is only applicable to the Ethernet interface because the serial links only allow one packet to transmit at a time.

5.3.4 Switch Board and Wiring

Since the virtual controllers now need to control and manage more devices, wiring modifications need to be considered accordingly to guarantee the feasibility of the proposed solution. Ethernet and serial links (such as RS-485 and RS-232) are commonly used in CPSs, so we consider both interfaces in this section.

Ethernet routers route packets between devices according to the IP addresses, hence no wiring modifications are needed for the Ethernet connection. On the other hand, when the devices are connected over the serial links, switch boards are used to facilitate the connection between the devices. A switch board acts as a networking device which multicasts one piece of data to all the ports which connect to the devices that need to receive it. Each level of virtualization needs at least one switch board. For example, as is shown in Fig. 5.5, the switch board 1 connects the VPLCs to the VTECs, and the switch board 2 at the field level connects the VTECs together with the sensors and the actuators. When a control packet is issued from the PLC 2-1 to an actuator in Group 5, the packet arrives at the switch board 1 from the "VPLC 1" port and is forwarded to all the VTEC ports on Row 3. All the VTECs receive the packet but only the TEC 5-1 and the TEC 5-2 will react and respond. Similarly, the Switch Board 2 will forward the packets from the TEC 5-1 and the

TEC 5-2 to the sensors and actuators in Group 5. The address field and the device ID in the packet header will be used to identify the destination device.

The use of switch boards can significantly minimize the number of cables in the network, especially for virtual controllers with a large VD. The switch board can be placed together with the virtual controllers.

5.4 Control Switching Strategy

Our solution assigns each controlled device with one PCI and multiple ACIs. The PCI is responsible for both sensing and actuation while the ACI is only used for data acquisition. If the PCI has been detected as anomalous, it will be removed from the network and an ACI will become the new PCI. Rather than acting as the cold standby, the ACIs act as the hot standby and continuously maintain an updated copy of the current object values on the controlled devices. When the ACI becomes the new PCI, it can directly apply control and actuation based on the object values stored on its local database. Compared to the cold standby, the host standby method can save a significant amount of control switch time because the new PCI do not need to reconstruct the system operating state through sending many data acquisition packets.

We consider two allocation strategies of the PCI: passive switching strategy and active switching strategy, as is discussed below.

5.4.1 Passive Switching Strategy

The controller instances that control the same set of devices are hosted on different physical controllers. The Server maintains the mapping between the controller instances and the physical controllers. The mapping is only changed when the PCI has been identified as anomalous and removed from the network. This allocation strategy is called the Passive Switching Strategy because the mapping is only changed after an attack has taken place.

For stealthy attackers who have access to the control network, it may not be difficult for them to infer the PCI allocation over time. They may sniff the network traffic, analyze the traffic volume and the application layer data payload using deep packet inspection, and infer the allocation of the PCI device. Since the passive switching strategy only switches the PCI after the attack has been detected, it is possible that some malicious actuation packets may have already been issued from the compromised PCI within the detection latency. The malicious actuation packets may have already been executed by the controlled devices before the malicious PCI is being removed from the network.

5.4.2 Active Switching Strategy

In the active switching strategy, the role of the PCI is dynamically switched among all instances of a controller. The Server maintains the mapping table to synchronize the PCI switching. Different from the passive switching strategy, this strategy takes a proactive defense approach and switches the control capability among all controller instances based on randomization. The randomization factor may include timestamp, CPU and memory utilization of individual physical controllers, security threat, and the Internet exposures of the controllers. The active switching strategy switches the PCI to random controllers at random timestamps, thus making the attackers more difficult to infer the current PCI from network traffic sniffing. This strategy can further minimize the impact of a potential compromise: the compromise of a PLC may not offer the attacker actuation capability. This idea is analogous to the Moving Target Defense (MTD) which has been used in the military defense. The active switching strategy turns the CPS architecture into a general switched structure. This strategy also requires all the ACIs to act as the hot standby, so the control switching overhead is minimal.

5.4.3 Discussion

5.4.3.1 Control Switch and System State

The control switching overhead is an important aspect to consider since the Server temporarily loses the control of the system during the switching time. Without continuous actuation, the system may operate in an unstable or a dangerous state. Different systems may behave differently when the continuous actuation signal is temporarily interrupted. For example, if a valve with a spring loses the actuation force, it will gradually return back to its original state (closed); however, if the valve is not attached with a spring, when the actuation force disappears, the valve will remain in the same position. When the new PCI controller is turned on, the system state is dependent on the last known operating state (before the control switching begins), the state transition speed and direction, and the switching latency. Hence, it is difficult to estimate the closed-form system state which is applicable for all types of CPSs.

However, we believe that the impact of the control switch time is minimal for a large number of CPS networks where the system response time is much larger than the control switching time. For example, the response time for a cooler or boiler varies from several minutes to a few hours, which is much larger than the control switch time (varies from several microseconds to milliseconds). In this case, the change of the system operating state during the control switching time is minimal. However, for CPS networks with faster system response time, such as the automobile control systems, the system state during the transition period needs to be further studied.

5.4.3.2 Limited Inter-Controllers Communication

If one controller instance is attacked, the Malware or virus on the malicious controller instance may spread and infect all other controller instances that control the same device. Different controller instances may collude to report false sensing values back to the Server

or apply malicious actuation signals to the controlled device. Therefore, we prevent the direct communication between the PCI and the ACIs of the same controller, and only allow the Server to directly communicate with each controller instance. We assume that the Server is equipped with adequate endpoint and network security protection and is safe from the cyberattacks. The limitation on the controller-to-controller communication facilitates the Server to identify the malicious controller instances using redundant sensing data from other controller instances.

5.4.3.3 Larger Attack Surface

The fat-tree structure allows each virtual controller to connect and communicate with multiple devices. If the hypervisor of a virtual controller is compromised, all controller instances that are managed by the hypervisor may be affected. In this case, if all these controller instances are the PCIs of different devices, all these controlled devices can be attacked. This will lead to a more dangerous situation and expose CPS with a larger attack surface compared to the current CPS architecture. To minimize the impact of the hypervior exploitation, we allocate the PCI of different devices to different virtual controllers and enforce deep packet inspection to detect actuation packets that are issued from the ACIs.

5.4.3.4 Different from Redundant Sensing

Both Section 4 and 5 propose to use sensing data replica to validate the data integrity and detect malicious controllers. However, Path Redundancy in Section 4 only focuses on the sensing part and is unable to switch the control when a controller is compromised. Our new CPS architecture takes care of control switching and turns the structure into a general switched structure. The introduce of virtualization further isolates different control contexts within a virtual controller, minimizing the attack impact of a compromise.

5.4.3.5 Scalability

The allocation of the controller instances is flexible and scalable. Depending on the application needs, the Server can create new controller instances, reallocate instances, or remove existing instances. When the current number of hardware controllers is insufficient to meet the redundancy requirement, the operator can install new devices and update the allocation mapping accordingly. This software-based allocation mechanism is suitable for a system with a growing number of controllers and networks with changing network topology.

5.5 Evaluation

The introduce of the new CPS architecture brings the following benefits: (1) it provides more data replicas to improve data validity and identify the malicious controllers; (2) it explores and utilizes the capability of the controllers to save the deployment cost; and (3) the software-based allocation mechanism of the virtual controllers is easy to maintain and update, and the mechanism is suitable for a network with a growing number of devices.

To evaluate the performance of the new architecture, we implemented the virtual controllers in an emulated Building Automation Network using Raspberry Pi devices. Since the real-world embedded controllers are more sophisticated than the Raspberry Pi devices in terms of the memory capacity and the computing capability, we expect the performance on the real-world embedded controllers to be better than that on the Raspberry Pi devices.

5.5.1 Emulation Settings

The testbed contains three components: the original PLC device, the virtual PLC (VPLC), and the TEC device, see Fig. 5.9. All three devices are emulated using the Raspberry Pi 3 Model B devices. Each device comes with 4 ARM Cortex-A53 cores @ 1.2GHz, 1 Gigabyte RAM, and 10/100 Ethernet. The PLC and VPLC are connected to

the TEC using the Ethernet respectively. The PLC-TEC channel simulates the original solution and the VPLC-TEC channel simulates our proposed solution. BACnet/IP is used as the communication protocol between the devices. The Ethernet bandwidth is 100 Mbps in our experiment.

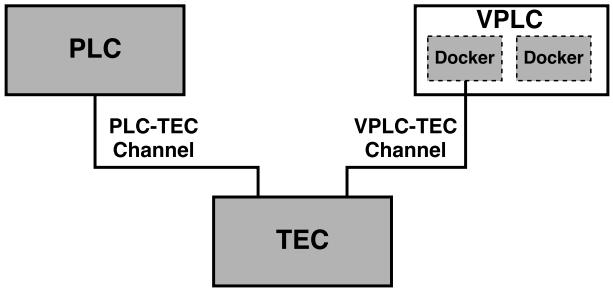


Figure 5.9: Emulated BAS network using Raspberry Pi devices.

Since the virtual machines are not supported on ARM processors, we implemented Docker [88] containers on the emulated VPLC to create virtual environments. Docker provides a lightweight OS-level virtualization solution which allows multiple independent containers to run independently within a physical device. We implemented Docker version 17.05.0-ce with the Raspbian GNU/Linux 8 (jessie) OS and the Kernel version 4.4.21-v7+.

A simple PID control program is running on the PLC and the Docker container within the VPLC. The control program periodically determines the actuation value based on the sensing input value, and the control sequence is as follows. First, the PLC sends a Read-Property Request to the TEC to query the present value of an object; second, the TEC looks up the target object in its database and replies with the current value; third, the PID program determines the actuation value based on the sensing input; next, the TEC constructs a WriteProperty Request which contains the actuation value and sends the packet to the TEC; finally, the TEC applies the actuation signal and replies the WriteProperty

request with an ACK. The control loop is executed every 500 ms in our emulation. The same control loop is executed on both the original PLC-TEC channel and the VPLC-TEC channel.

5.5.2 Emulation Results

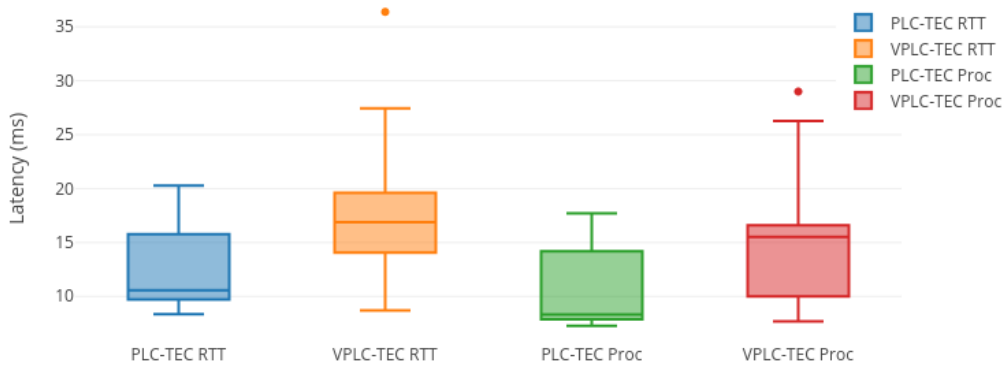


Figure 5.10: The box plot of the RTT and the processing latency in both channels.

	PLC-TEC (ms)	VPLC-TEC (ms)	% INC
avgRTT	11.98	16.37	36.64
avgProc	10.02	14.35	43.21
DIFF	1.96	2.02	3.06

Table 5.1: Timing Performance of two channels.

We measured the latency, CPU utilization, memory usage, and network traffic overhead to evaluate the performance of our solution. Fig. 5.10 demonstrates the box plot

of the RTT and the processing latency distribution in both channels. The aggregated and average results are shown in Table 5.1. The average RTT is 11.98 ms in the PLC-TEC channel and 16.37 ms in the VPLC-TEC channel; and the average processing latency is 10.02 ms in the PLC-TEC channel and 14.35 ms in the VPLC-TEC channel. The difference between the avgRTT and avgRroc (*i.e.* DIFF) includes the propagation delay of the four packets and the processing latency at the TEC. The DIFF is 1.96 ms and 2.02 ms in the PLC-TEC and the VPLC-TEC channel, respectively.

When the control program is executing, the CPU utilization is 1.0% on the original PLC device, and is 1.6% on the VPLC with one Docker container running; the memory usage is 6.6% on the original PLC device, and is 8.8% on the VPLC device with one Docker container running. Since the VPLC-TEC channel does not generate additional network traffic, so the network traffic overhead is zero.

From the latency measurement in the Ethernet interface, we estimate the propagation delay on the RS-485 serial link which is commonly used in the field level CPS networks. The typical bandwidth of the RS-485 is around 50 Mbps. We assume that the average BACnet packet is 100 bytes, then the propagation delay of four packets is 0.064 ms. From the DIFF value in Table 5.1, we can estimate the processing latency at the TEC is around 1.928 ms. Therefore, the estimated avgRTT for the PLC-TEC channel using the RS-485 links is 12.01 ms; and the estimated avgRTT for the VPLC-TEC channel using the RS-485 links is 16.34 ms.

5.5.3 Virtualization Degree

5.5.3.1 Ethernet

The Virtualization Degree (VD) of a physical controller is dependent on three constraints: *Timing Factor (TF)*, *CPU Factor (CF)*, and *Memory Factor (MF)*, and is bounded

by the tightest constraint among the three: $VD \leq min\{TF, CF, MF\}$. TF is dependent on the capability of the controller and the control frequency.

From the emulation results in Table 5.1, the avgRTT in the VPLC-TEC channel in the Ethernet is 16.37 ms. A VPLC with 100 ms control period is able to host at most:

$$\begin{aligned} VD &\leq min\{TF, CF, MF\} \\ &= min\{\frac{100}{16.37}, \frac{1}{1.6\%}, \frac{1}{8.8\%}\} \\ &= min\{6.11, 62.50, 11.36\} = 6.11 \end{aligned} \tag{5.4}$$

A VPLC with 1 second control period is able to host at most:

$$\begin{aligned} VD &\leq min\{TF, CF, MF\} \\ &= min\{\frac{1000}{16.37}, \frac{1}{1.6\%}, \frac{1}{8.8\%}\} \\ &= min\{61.09, 62.50, 11.36\} = 11.36 \end{aligned} \tag{5.5}$$

Therefore, each emulated VPLC using the Raspberry Pi devices (with 1G RAM) is able to run 6 to 11 controller instances concurrently. The real-world embedded controllers are expected to host more controller instances than the Raspberry Pi devices.

5.5.3.2 Serial Links

RS-485 supports the multidrop communication, which means only one message can be transmitted over the link at a time. In addition to the constraints described for the Ethernet interface, we also need to consider the *Link Factor (LF)* constraint of the serial links: the VD is also bounded by the number of packets that can be transmitted over the link within

a control period:

$$LF = \frac{Control\ Period}{Propagation\ Delay\ for\ Packets\ Per\ Period} \tag{5.6}$$

Since the propagation delay for four packets in RS-485 is 0.064 ms, the VD for 100 ms control period in the RS-485 is:

$$\begin{aligned} VD &\leq min\{TF, CF, MF, LF\} \\ &\leq min\{\frac{100}{16.37}, \frac{1}{1.6\%}, \frac{1}{8.8\%}, \frac{100}{0.064}\} \\ &= min\{6.11, 62.50, 11.36, 1562.50\} = 6.11 \end{aligned} \tag{5.7}$$

The VD for 1 second control period in the RS-485 is:

$$\begin{aligned} VD &\leq min\{TF, CF, MF, LF\} \\ &\leq min\{\frac{1000}{16.37}, \frac{1}{1.6\%}, \frac{1}{8.8\%}, \frac{1000}{0.064}\} \\ &= min\{61.09, 62.50, 11.36, 15625.00\} = 11.36 \end{aligned} \tag{5.8}$$

From Eq. 5.4 - 5.8, we conclude that with the current control period, the VD in the RS-485 interface is the same as the VD in the Ethernet interface.

5.6 Summary

Cyber-Physical Systems contain multiple embedded controllers which may become the Single Point of Failures (SPOFs): the compromise of one controller will affect the devices which are controlled by it. We introduce a new architecture of CPS to overcome SPOFs without requiring additional hardware devices. Our solution changes the CPS structure

into a general switched structure. The redundant controller instances provide data replica and control backups when one controller instance is attacked. We discussed both the passive switching strategy and the active switching strategy. We evaluate the performance of VPLCs using Raspberry Pi devices and Docker containers. Our result demonstrates that with $35 Raspberry Devices, we are able to run 6-11 control programs concurrently. The result would be better for real-world embedded controllers with better computing capability and memory capacity. Therefore, this architecture should be adopted by the industry as it provides better security protection with minimal cost.

6. CONCLUSION

Cyber-physical Systems (CPSs) are ubiquitous in critical infrastructures such as power systems, water treatment plants, nuclear facilities, and transportation networks. The safety and security of CPSs have recently become a crucial research topic. In this work, we develop solutions to protect CPS networks (with a focus on Building Automation Networks) against cyberattacks from four perspectives: using anomaly detection to identify suspicious network traffic, improving system resilience through commensurate response, improving data validity through path redundancy, and introducing fat-tree structure to overcome Single-Point-of-Failures.

To develop an anomaly detector for Building Automation Networks, we collect BACnet traffic from a BAS at various vantage points. We conduct in-depth statistical analysis at aggregated level and individual flow-service level to understand traffic behavior at different types of end hosts. Our analysis reveals that (i) aggregated BACnet traffic does not exhibit diurnal patterns nor look strictly periodic because it consists of time-driven messages with different periodic behavior as well as non-periodic streams; and (ii) the non-periodic traffic includes human-driven and event-driven traffic. Such traffic behavior has not been observed in previous CPS network traffic analysis. Based on the traffic patterns, we construct flow-service models for the time-driven traffic and develop THE-Driven Anomaly Detector which adopts different mechanisms for each category of traffic. We evaluate the anomaly detector using k-fold cross validation and synthetic attacks. The proposed THE-Driven Anomaly Detector is shown to be able to effectively detect suspicious traffic in BAS networks with small false alarm rate.

Next, we focus on developing solutions to improve system resilience, minimizing the impact of potential attacks which aim to drive the system into a critical state. In Section 3,

we propose adaptive Commensurate Response (CR) as a cross-domain technique that protects a control system through appropriate control parameter selection against attacks coming from the communication domain. We provide detailed study on both change-driven CR and criticality-driven CR. Our case study on automobile cruise control demonstrates that change-driven CR can be used against setpoint attacks and criticality-driven CR is effective to combat both setpoint and actuation attacks. CR can effectively improve the system resilience and attack survivability and facilitate other defense mechanisms such as firewalls and IDS to better protect CPS.

CPSs employ a number of distributed embedded controllers (PLCs and RTUs) to control and mange the network. The embedded controllers normally follow a hierarchical tree structure, and may become the Single-Point-of-Failures of the network. The compromise of one controller will affect the devices which are controlled by the affected controller. Section 4 shows that Path Redundancy is a simple and effective solution to counter compromised PLCs in CPS. The proposed redundant paths are enabled with existing PLCs and are only used for sensing purposes. The emulation results have shown that our solution could effectively prevent data integrity attacks and detect false command attacks from a single source. Our solution is cost-efficient, easy-to-deploy, and suitable for different types of CPS with different communication interfaces.

In Section 5, we introduce a new CPS architecture (fat-tree structure) which is able to overcome SPOFs without requiring additional hardware devices. The solution provides data replica to detect malicious embedded controllers and enables control switching when a controller is identified as anomalous. The introduce of virtualization takes advantage of the capacity of existing controllers to host multiple controller instances. We discuss both the passive switching strategy and the active switching strategy. We emulate the VPLCs using Raspberry Pi devices and Docker containers. It is shown that the Raspberry Pi is able to concurrently run 6-11 virtual controllers. Our proposed solution should be practical and

feasible for the real-world CPS controllers with better computing capability and memory capacity. This architecture should be adopted by the industry as it provides better security protection with minimal cost.

6.1 Future Work

The anomaly detector of our BACnet dataset has good detection accuracy and latency. In future work, we plan to test our anomaly detector with new datasets. In addition, we are interested to implement the adaptive Commensurate Response to other types of CPSs with additional QoS requirements. In addition to the PID controllers used in this work, we can also combine CR with other types of controllers. We are also interested to evaluate the performance of CR with real-world applications and combine CR with other CPS defense mechanisms such as Dynamic Watermarking to better protect CPSs.

REFERENCES

[1] X. Yu and Y. Xue, "Smart grids: A cyber–physical systems perspective," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1058–1070, 2016.

[2] Wikipedia, "Building automation — wikipedia, the free encyclopedia," 2017. https://en.wikipedia.org/w/index.php?title=Building_automation&oldid=776185496.

[3] Wikipedia, "Sewage treatment — wikipedia, the free encyclopedia," 2017. https://en.wikipedia.org/w/index.php?title=Sewage_treatment&oldid=777151381.

[4] "1815-2012 - IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3)," *IEEE Std. 1815-2012*, 2012.

[5] I. Modbus Organization, "Modbus home page," http://www.modbus.org/.

[6] ASHRAE, "Standard 135-2012 – bacnet–a data communication protocol for building automation and control networks (ANSI approved)," *ASHRAE135-2012*, 2012.

[7] B. A. Group, "Bacnet website," http://www.bacnet.org/.

[8] Wikipedia, "Stuxnet — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=Stuxnet&oldid=750104845.

[9] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *SANS Indus. Control Sys.*, 2016.

[10] B. Swan, "Building wide-area networks with bacnet (part 2)," http://www.bacnet.org/Bibliography/ES-7-99/IPPART2.HTM.

[11] WhiteScope, "Understanding building automation system exposures - october 2015," 2015. https://whitescope.io/smartbuildingsecurity/.

[12] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by Internet-wide scanning," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security (CCS'15)*, ACM, Oct 2015.

[13] Shodan, "Shodan search engine," http://www.shodan.io/.

[14] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[15] W. J. Broad and D. E. Sanger, "Worm was perfect for sabotaging centrifuges," *New York Times*, vol. 18, 2010.

[16] R. R. R. Barbosa, R. Sadre, and A. Pras, "A first look into scada network traffic," in *Network Operations and Management Symposium*, pp. 518–521, IEEE, 2012.

[17] R. R. R. Barbosa, R. Sadre, and A. Pras, "Towards periodicity based anomaly detection in scada networks," IEEE Industrial Electronics Society, 2012.

[18] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *International Journal of Critical Infrastructure Protection (IJCIP)*, vol. 6, no. 2, pp. 63–75, 2013.

[19] A. Kleinmann and A. Wool, "Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensic," *Journal of Digital Forensics, Security and Law*, vol. 9, no. 2, pp. 37–50, 2014.

[20] R. R. R. Barbosa, R. Sadre, and A. Pras, "Difficulties in modeling scada traffic: a comparative analysis," in *Passive and Active Measurement*, pp. 126–135, Springer, 2012.

[21] R. Krejčí, P. Čeleda, and J. Dobrovolnỳ, "Traffic measurement and analysis of building automation and control networks," in *Dependable Networks and Services*, pp. 62–73, Springer, 2012.

[22] H. M. Newman, "Broadcasting bacnet," *ASHRAE Journal*, 2010. `http://www.bacnet.org/Bibliography/BACnet-Today-10/Newman_2010.pdf`.

[23] S. T. Bushby, "Bacnettm: a standard communication infrastructure for intelligent buildings," *Automation in Construction*, vol. 6, no. 5-6, pp. 529–540, 1997.

[24] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*, pp. 75–86, ACM, 2003.

[25] Wireshark `https://www.wireshark.org/`.

[26] G. Dewaele, Y. Himura, P. Borgnat, K. Fukuda, P. Abry, O. Michel, R. Fontugne, K. Cho, and H. Esaki, "Unsupervised host behavior classification from connection patterns," *International Journal of Network Management*, vol. 20, no. 5, pp. 317–337, 2010.

[27] R. R. R. Barbosa, "Anomaly detection in scada systems-a network based approach," 2014.

[28] J. Bigham, D. Gamez, and N. Lu, "Safeguarding scada systems with anomaly detection," in *Computer Network Security*, pp. 171–182, Springer, 2003.

[29] W. Gao, T. Morris, B. Reaves, and D. Richey, "On scada control system command and response injection and intrusion detection," in *eCrime Researchers Summit (eCrime), 2010*, pp. 1–9, IEEE, 2010.

[30] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer, "Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol," in *Proceedings of Annual Cyber Security and Information Intelligence Research Workshop*, p. 5, ACM, 2013.

[31] C.-H. Tsang and S. Kwong, "Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction," in *IEEE International Conference on Industrial Technology (ICIT'05)*, pp. 51–56, IEEE, 2005.

[32] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for scada systems," in *Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies (NPIC&HMIT'05)*, pp. 12–16, Citeseer, 2006.

[33] M. N. Johnstone, M. Peacock, and J. den Hartog, "Timing attack detection on bacnet via a machine learning approach," 2015.

[34] P. Čeleda, R. Krejčí, and V. Krmíček, "Flow-based security issue detection in building automation and control networks," in *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, pp. 64–75, Springer, 2012.

[35] Z. Pan, S. Hariri, and Y. Al-Nashif, "Anomaly based intrusion detection for building automation and control networks," in *2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA)*, pp. 72–77, IEEE, 2014.

[36] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in scada systems," *IEEE Transactions on Industrial Informatics*, vol. 7, pp. 179–186, May 2011.

[37] I. N. Fovino, A. Carcano, T. De Lacheze Murel, A. Trombetta, and M. Masera, "Modbus/dnp3 state-based intrusion detection system," in *IEEE International Conference on Advanced Information Networking and Applications (AINA'10)*, pp. 729–736, IEEE, 2010.

[38] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46, pp. 1–12, Citeseer, 2007.

[39] A. Valdes and S. Cheung, "Communication pattern anomaly detection in process control systems," in *IEEE Conference on Technologies for Homeland Security (HST '09)*, pp. 22–29, IEEE, 2009.

[40] A. Valdes and S. Cheung, "Intrusion monitoring in process control systems," in *Hawaii International Conference on System Sciences (HICSS'09)*, pp. 1–7, IEEE, 2009.

[41] J. Verba and M. Milvich, "Idaho national laboratory supervisory control and data acquisition intrusion detection system (scada ids)," in *IEEE Conference on Technologies for Homeland Security (HST'08)*, pp. 469–473, IEEE, 2008.

[42] R. R. R. Barbosa and A. Pras, "Intrusion detection in scada networks," in *Mechanisms for Autonomous Management of Networks and Services*, pp. 163–166, Springer, 2010.

[43] C. R. Taylor, C. A. Shue, and N. R. Paul, "A deployable scada authentication technique for modern power grids," in *IEEE International Energy Conference (ENERGYCON'14)*, IEEE, 2014.

[44] Z. Zheng and A. L. N. Reddy, "Towards improving data validity of cyber-physical systems through path redundancy," in *Proceedings of the 3rd ACM Workshop on*

Cyber-Physical System Security (CPSS'17), pp. 91–102, ACM, 2017.

[45] D. Germanus, A. Khelil, and N. Suri, "Increasing the resilience of critical scada systems using peer-to-peer overlays," in *Architecting Critical Systems*, pp. 161–178, Springer, 2010.

[46] B. Vaidya, D. Makrakis, and H. T. Mouftah, "Authentication and authorization mechanisms for substation automation in smart grid network," *IEEE Network*, vol. 27, no. 1, pp. 5–11, 2013.

[47] A. K. Wright, J. A. Kinast, and J. McCarty, "Low-latency cryptographic protection for scada communications," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'04)*, pp. 263–277, Springer, 2004.

[48] B. Satchidanandan and P. Kumar, "Dynamic watermarking: Active defense of netwk. cyber-physical systems," *Proceedings of the IEEE*, 2016.

[49] W.-H. Ko, B. Satchidanandan, and P. Kumar, "Theory and impl. of dynamic watermarking for cybersecurity of advanced trans. syst.," in *IEEE Conference on Communications and Network Security (CNS'16)*, pp. 416–420, IEEE, 2016.

[50] S. Weerakkody, Y. Mo, and B. Sinopoli, "Detecting integrity attacks on control systems using robust physical watermarking," in *IEEE Decision and Control (CDC'14)*, pp. 3757–3764, IEEE, 2014.

[51] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[52] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *IEEE International Conference on Smart Grid Communication (SmartGridComm'10)*, pp. 226–231, IEEE, 2010.

[53] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, 2012.

[54] C. Kwon, W. Liu, and I. Hwang, "Security analysis for cyber-physical systems against stealthy deception attacks," in *American Control Conference (ACC'13)*, pp. 3344–3349, IEEE, 2013.

[55] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *47th Ann. Allerton Conf. on Comm. Control, and Compt. (Allerton'09)*, pp. 911–918, IEEE, 2009.

[56] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *International Conference on Distributed Computing Systems Workshops (ICDCS Workshop'08)*, pp. 495–500, IEEE, 2008.

[57] A. Manocha and A. Sharma, "Three axis aircraft autopilot control using genetic algorithms: An experimental study," in *IEEE International Advance Computing Conference (IACC'09)*, pp. 171–174, IEEE, 2009.

[58] J. Ziegler and N. Nichols, "Optimum settings for automatic controllers," *ASME DC*, vol. 115, no. 2B, pp. 220–222, 1993.

[59] M. L. Luyben and W. L. Luyben, *Esse. of proc. ctrl.* McGraw-Hill, 1997.

[60] S. V. Labs, "Designing of reconfigurable architecture for pid controller," 2017. `http://coep.vlab.co.in/index.php?sub=29&brch=88&sim=1312&cnt=1`.

[61] P. Ioannou, Z. Xu, S. Eckert, D. Clemons, and T. Sieja, "Intelligent cruise control: theory and experiment," in *IEEE Decision and Control (CDC'93)*, pp. 1885–1890, IEEE, 1993.

[62] R. Rajamani, *Vehicle dynamics and control*. Science & Business Media, 2011.

[63] D.-T. HENTUNEN, "Control tutorials for matlab & simulink," June 2014. http://ctms.engin.umich.edu/CTMS/index.php?aux=Home.

[64] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *25th USENIX Security Symposium (USENIX Security'16)*, pp. 911–927, USENIX Association, 2016.

[65] Y. Mo, T. H.-J. Kim, K. Brancik, D. Dickinson, H. Lee, A. Perrig, and B. Sinopoli, "Cyber–physical security of a smart grid infrastructure," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 195–209, 2012.

[66] Z. Zheng and A. L. N. Reddy, "Safeguarding building automation networks: The-driven anomaly detector based on traffic analysis," in *The 26th International Conference on Computer Communications and Networks (ICCCN'17)*, IEEE, 2017.

[67] S. L. Keoh, K. W. K. Au, and Z. Tang, "Securing industrial control system: An end-to-end integrity verification approach," in *Proceedings Industrial Control System Security Workshop (ICSS'15)*, ACSA, 2015.

[68] Y. Mo and B. Sinopoli, "Integrity attacks on cyber-physical systems," in *Proceedings of the 1st international conference on High Confidence Networked Systems*, pp. 47–54, ACM, 2012.

[69] M. Krotofil and A. A. Cárdenas, "Resilience of process control systems to cyber-physical attacks," in *Nordic Conference on Secure IT Systems*, pp. 166–182, Springer, 2013.

[70] R. Langner, "To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve," *The Langner Group*, 2013.

[71] NCCIC/ICS-CERT, "Alert (ir-alert-h-16-056-01): Cyber-attack against ukrainian critical infrastructure," 2016. https://ics-cert.us-cert.gov/alerts/

IR-ALERT-H-16-056-01.

[72] B. Cai, Y. Liu, Z. Liu, F. Wang, X. Tian, and Y. Zhang, "Development of an automatic subsea blowout preventer stack control system using plc based scada," *ISA transactions*, vol. 51, no. 1, pp. 198–207, 2012.

[73] Z. Xie, G. Manimaran, V. Vittal, A. Phadke, and V. Centeno, "An information architecture for future power systems and its reliability analysis," *IEEE Transactions on Power Systems*, vol. 17, no. 3, pp. 857–863, 2002.

[74] Q. Zhang, T. Yu, and P. Ning, "A framework for identifying compromised nodes in sensor networks," in *Securecomm and Workshops, 2006*, pp. 1–10, IEEE, 2006.

[75] B. Zhu and S. Sastry, "Scada-specific intrusion detection/prevention systems: a survey and taxonomy," in *Proceedings of the 1st Workshop on Secure Control Systems (SCS'10)*, 2010.

[76] Z. Alliance *et al.*, "Zigbee specification," 2006.

[77] S. ASHRAE, "Standard 135-1995: Bacnet-a data communication protocol for building automation and control networks," *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Atlanta, Georgia, USA*, 1995.

[78] M. E. Hazen, "Understanding some basic recommended standards for serial data communications-a comparison of rs-232, rs-422 and rs-485," *http://www. intersil. com/data/wp/WP0585. pdf*, 2003.

[79] Wikipedia, "Rs-485 — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=RS-485&oldid=749971651.

[80] B. Electronics, "Rs-485 tips, tricks, questions answers.," 2016. http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/

RS-485-Tips,-Tricks,-Questions-Answers/RS-485-Tips,-Tricks,-Questions-Answers.pdf.

[81] S. Karg, "Bacnet stack: An open source bacnet protocol stack for embedded systems," 2015. http://bacnet.sourceforge.net/.

[82] H. Beitollahi and G. Deconinck, "Analyzing the chord peer-to-peer network for power grid applications," in *Fourth IEEE Young Researchers Symposium in Electrical Power Engineering*, p. 5, 2008.

[83] G. Deconinck, T. Rigole, H. Beitollahi, R. Duan, B. Nauwelaers, E. Van Lil, J. Driesen, R. Belmans, and G. Dondossola, "Robust overlay networks for microgrid control systems," in *Proc. Workshop on Architecting Dependable Systems (WADS '07)*, pp. 148–153, 2007.

[84] H. Gjermundrod, D. E. Bakken, C. H. Hauser, and A. Bose, "Gridstat: A flexible qos-managed data dissemination framework for the power grid," *IEEE Transactions on Power Delivery*, vol. 24, no. 1, pp. 136–143, 2009.

[85] A. Khelil, D. Germanus, and N. Suri, "Protection of scada communication channels," in *Critical Infrastructure Protection*, pp. 177–196, Springer, 2012.

[86] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[87] D. Germanus, A. Khelil, J. Schwandke, and N. Suri, "Coral: Reliable and low-latency p2p convergecast for critical sensor data collection," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm'13)*, pp. 300–305, IEEE, 2013.

[88] S. Hykes, "Docker website," March 2013. https://www.docker.com/.